In [1]:
```python
import re
```

Question 1- Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

In [2]:
```python
# Solution
def specific_char(string):
    pattern = re.compile(r'[^a-zA-Z0-9.]')
    string = pattern.search(string)
    return not bool(string)

print(specific_char("ABCDEFabcdef123450"))
print(specific_char("*&%@#!}{"))
```

```
True
False
```

In pattern the caret is used only to match the pattern at the beginning of the string which starts with either a-z or a-z or 0-9 but not the new line character.

In [ ]:
```python

```

Question 2- Create a function in python that matches a string that has an a followed by zero or more b's.

In [3]:
```python
# Solution
def text_match(text):
        patterns = 'ab*'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("data"))
print(text_match("abacus"))
print(text_match("abbc"))
```

```
Found a match!
Found a match!
Found a match!
```

In pattern the asterisk is used only to match zero or more occurence of b.

In [ ]:
```python

```

Question 3- Create a function in python that matches a string that has an a followed by one or more b's.

In [4]:
```python
# Solution
def text_match(text):
        patterns = 'ab+'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("a"))
print(text_match("abc"))
```

```
Not matched!
Found a match!
```

In pattern the plus is used only to match one or more occurence of b.

```
In [ ]:
```

Question 4- Create a function in Python and use RegEx that matches a string that has an a followed by zero or one b.

```
In [5]: def text_match(text):
                patterns = 'ab?.'
                if re.search(patterns,  text):
                        return 'Found a match!'
                else:
                        return('Not matched!')

        print(text_match("ab"))
        print(text_match("abc"))
        print(text_match("abbc"))
        print(text_match("aabbc"))
        print(text_match("aabbc"))
```

```
Found a match!
Found a match!
Found a match!
Found a match!
Found a match!
```

In pattern the ? is used only to match zero or one occurence and dot is used to match any character except a newline character after b.

```
In [ ]:
```

Question 5- Write a Python program that matches a string that has an a followed by three 'b'

```
In [6]: def text_match(text):
                patterns = 'ab{3}'
                x= re.findall(patterns,  text)
                if x:
                        print(x)
                        return 'Found a match!\n'
                else:
                        print(x)
                        return('Not matched!\n')

        print(text_match("Hello!, This is abbb"))
        print(text_match("Hello!, This is abbbbbc"))
        print(text_match("Hello!, This is abbc"))
```

```
['abbb']
Found a match!

['abbb']
Found a match!

[]
Not matched!
```

In pattern the {3} is used only to match a string that has an a followed by three 'b'.

```
In [ ]:
```

Question 6- Write a regular expression in Python to split a string into uppercase letters.

Sample text: “ImportanceOfRegularExpressionsInPython”

Output: [‘Importance’, ‘Of’, ‘Regular’, ‘Expression’, ‘In’, ‘Python’]

In [7]:
```
text= "ImportanceOfRegularExpressionsInPython"
upper_case = [s for s in re.split("([A-Z][^A-Z]*)", text) if s]

print(upper_case)
```
```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [8]:
```
#Solution 2


text= "ImportanceOfRegularExpressionsInPython"

upper_case = [s for s in re.split("([A-Z][^A-Z]*)", text) if s]
print(upper_case)
```
```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [ ]:

Question 7- Write a Python program that matches a string that has an a followed by two to three 'b'.

In [9]:
```
def text_match(text):
        patterns = 'ab{2,3}'
        x= re.findall(patterns,  text)
        if x:
                print(x)
                return 'Found a match!\n'
        else:
                return('Not matched!')

print(text_match("ab"))
print(text_match("aabbbbbc abb"))
```
```
Not matched!
['abbb', 'abb']
Found a match!
```

In [ ]:

Question 8- Write a Python program to find sequences of lowercase letters joined with a underscore.

In [10]:
```
def text_match(text):
        patterns ='^[a-z]+_[a-z]+'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("data_scientist"))
print(text_match("Data_scientist"))
print(text_match("data_Scientist"))
```
```
Found a match!
Not matched!
Not matched!
```

In [11]:
```
def text_match(text):
        patterns ='^[a-z]+_[a-z]+'
        x=re.findall(patterns,  text)
        if x:
                print(x)
                return 'Found a match! \n'
        else:
                return('Not matched! \n')

print(text_match("data_scientist"))
print(text_match("Data_scientist"))
print(text_match("data_Scientist"))
```

```
['data_scientist']
Found a match!

Not matched!

Not matched!
```

In [ ]:

Question 9- Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'.

In [12]:
```
def text_match(text):
        patterns = 'a.*b$'
        x=re.findall(patterns,  text)
        if x:
                print(x)
                return 'Found a match!\n'
        else:
                return('Not matched!\n')

print(text_match("This is axyzab"))
print(text_match("aabAbbbc"))
print(text_match("accddbbjjjb"))
```

```
['axyzab']
Found a match!

Not matched!

['accddbbjjjb']
Found a match!
```

In [13]:
```
def text_match(text):
        patterns = 'a\w*b$'
        x=re.search(patterns,  text)
        if x:
                print(x)
                print(x.group())
                return 'Found a match!\n'
        else:
                return('Not matched!\n')

print(text_match("This is aabbbdb"))
print(text_match("aabAbbbc"))
print(text_match("accddbbjjjb"))
```

```
<re.Match object; span=(8, 15), match='aabbbdb'>
aabbbdb
Found a match!
```

```
Not matched!

<re.Match object; span=(0, 11), match='accddbbjjjb'>
accddbbjjjb
Found a match!
```

In [ ]:

Question 10- Write a Python program that matches a word at the beginning of a string.

In [14]:

```python
def text_match(text):
        patterns = '^\w+'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match(" The quick brown fox jumps over the lazy dog."))
print(text_match("@The quick brown fox jumps over the lazy dog."))
```

```
Found a match!
Not matched!
Not matched!
```

In [15]:

```python
def text_match(text):

        patterns = '^@'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match(" The quick brown fox jumps over the lazy dog."))
print(text_match("@The quick brown fox jumps over the lazy dog."))
```

```
Not matched!
Not matched!
Found a match!
```

In [ ]:

Question 11- Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.

In [16]:

```python
# Solution
def text_match(text):
        patterns = '[a-zA-Z]+[0-9_]+'
        x=re.findall(patterns,  text)
        if x:
                print(x)
                return 'Found a match!\n'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog named xYz_123."))
print(text_match("Python_Exercises_1"))
```

```
['xYz_123']
Found a match!

['Python_', 'Exercises_1']
```

```
Found a match!
```

In [17]:
```
def text_match(text):
        patterns = '[a-zA-Z]+[0-9_]+|[0-9_]+[a-zA-Z]+'
        x=re.search(patterns,  text)
        if x:
                print(x)
                print(x.group())
                return 'Found a match!\n'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog named XyZ_123."))
print(text_match("The quick brown fox jumps over the lazy dog named 123_XyZ."))
print(text_match("Python_Exercises_1"))
```

```
<re.Match object; span=(50, 57), match='XyZ_123'>
XyZ_123
Found a match!

<re.Match object; span=(50, 57), match='123_XyZ'>
123_XyZ
Found a match!

<re.Match object; span=(0, 7), match='Python_'>
Python_
Found a match!
```

In [ ]:

Question 12- Write a Python program where a string will start with a specific number.

In [18]:
```
# Solution
def match_num(string):
    text = re.compile(r"^5")
    if text.match(string):
        return True
    else:
        return False
print(match_num('5-2345861'))
print(match_num('6-2345861'))
```

```
True
False
```

In [19]:
```
def match_num(string):
    x=re.findall(r'\b5[\w]*', string)
    if x:
        return True
    else:
        return False
print(match_num('5-dataScience123'))
print(match_num('6_dataScience123'))
```

```
True
False
```

\b5[\w]asterisk- In this pattern \b5 is used to return a string starting with 5 and [\w]* is used to return zero or more occurrence of any word characters (characters from a to Z, digits from 0-9, and the underscore _ character). So this pattern will return a string starting with 5 followed by any word.

In [ ]:

Question 13- Write a Python program to remove leading zeros from an IP address

In [20]:
```
ip = "216.08.094.196"
string = re.sub('\.[0]*', '.', ip)
print(string)
```

```
216.8.94.196
```

In [21]:
```
ip = "216.008.094.196"
string = re.sub('\.[0]*', '.', ip)
print(string)
```

```
216.8.94.196
```

\.[0]asterisk- In this pattern '\.' is used to find the string starting with dot and [0]* is used to find zero or more occurrence of zero. So this code will replace the dot followed by any number of zeroes with a dot.

In [ ]:
```

```

Question 14- Write a regular expression in python to match a date string in the form of Month name followed by day number and year stored in a text file.

Sample text : On August 15th 1947 that India was declared independent from British colonialism, and the reins of control were handed over to the leaders of the Country’.

Output- August 15th 1947

In [22]:
```
string= "On August 15th 1947 that India was declared independent from British colonialis
pattern = "([a-zA-Z]+) (\d+[a-z]+) (\d+)"

matched = re.search(pattern, string)
print ("Output: %s" % (matched.group()))
```

```
Output: August 15th 1947
```

([a-zA-Z]+)- It is used to return the one or more occurence of character alphabetically between a and z, lower case or upper case.

(\d+[a-z]+)- In this pattern \d+ is used to find one or more occurrences of digits and [a-z]+ is used to find one or more occurrences of lowercase letters between a to z. Therefore this pattern will return the digits followed by lowercase alphabetic characters.

(\d+)- In this pattern \d+ is used to find one or more occurence of digits.

This pattern will return the date string in the form of Month name followed by day number and year

In [ ]:
```

```

Question 15- Write a Python program to search some literals strings in a string.

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox', 'dog', 'horse'

In [23]:
```
patterns = [ 'fox', 'dog', 'horse' ]
text = 'The quick brown fox jumps over the lazy dog.'
for pattern in patterns:
    print('Searching for "%s" in "%s" ->' % (pattern, text))
    if re.search(pattern,  text):
        print('Matched!\n')
    else:
        print('Not Matched!')
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog." ->
Matched!

Searching for "dog" in "The quick brown fox jumps over the lazy dog." ->
Matched!

Searching for "horse" in "The quick brown fox jumps over the lazy dog." ->
Not Matched!
```

In [24]:
```
patterns = [ 'fox', 'dog', 'horse' ]
text = 'The quick brown fox jumps over the lazy dog.'

for pattern in patterns:
    print('Searching for "%s" in "%s" ' % (pattern, text))
    x=re.findall(pattern,  text)
    if x:
        print(x)
        print('Literal Found\n')
    else:
      print('Literal Not Found')
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog."
['fox']
Literal Found

Searching for "dog" in "The quick brown fox jumps over the lazy dog."
['dog']
Literal Found

Searching for "horse" in "The quick brown fox jumps over the lazy dog."
Literal Not Found
```

In [25]:
```
def literals(text, patterns):

  for pattern in patterns:
      print('Searching for "%s" in "%s" ' % (pattern, text))
      x=re.findall(pattern,  text)
      if x:
          print(x)
          print('Literal Found\n')
      else:
          print('Literal Not Found')

print(literals('The quick brown fox jumps over the lazy dog.', patterns = [ 'fox', 'dog'
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog."
['fox']
Literal Found

Searching for "dog" in "The quick brown fox jumps over the lazy dog."
['dog']
Literal Found

Searching for "horse" in "The quick brown fox jumps over the lazy dog."
Literal Not Found
None
```

In [ ]:

Question 16- Write a Python program to search a literals string in a string and also find the location within the original string where the pattern occurs

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox'

In [26]:
```
pattern = 'fox'
text = 'The quick brown fox jumps over the lazy dog.'
match = re.search(pattern, text)
s = match.start()
e = match.end()
print('Found "%s" in "%s" from %d to %d ' % (match.re.pattern, match.string, s, e))
```

```
Found "fox" in "The quick brown fox jumps over the lazy dog." from 16 to 19
```

Here start() returns the index of the start of the substring matched by regex.search() and end() returns the index of the end of the substring matched by regex.search().

In [ ]:

Question 17- Write a Python program to find the substrings within a string.

Sample text : 'Python exercises, PHP exercises, C# exercises'

Pattern : 'exercises'

Note: There are two instances of exercises in the input string.

In [27]:
```
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
for match in re.findall(pattern, text):
    print('Found "%s"' % match)
```

```
Found "exercises"
Found "exercises"
Found "exercises"
```

In [ ]:

Question 18- Write a Python program to find the occurrence and position of the substrings within a string.

In [28]:
```
# Solution
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
for match in re.finditer(pattern, text):
    s = match.start()
    e = match.end()
    print('Found "%s" at %d:%d' % (text[s:e], s, e))
```

```
Found "exercises" at 7:16
Found "exercises" at 22:31
Found "exercises" at 36:45
```

In [ ]:

Question 19- Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format.

In [29]:
```
# Solution
def change_date_format(dt):
        return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
dt1 = "2026-01-02"
print("Original date in YYY-MM-DD Format: ",dt1)
print("New date in DD-MM-YYYY Format: ",change_date_format(dt1))
```

```
Original date in YYY-MM-DD Format:  2026-01-02
New date in DD-MM-YYYY Format:  02-01-2026
```

(\d{4})- It is a group to find the numbers (0-9) of length between 4 in a given string.

(\d{1,2})-It is a group to find the numbers (0-9) of length between 1 to 2 in a given string.

(\d{1,2})- It is a group to find the numbers (0-9) of length between 1 to 2 in a given string.

\\3- Used to replace year to day. This means that group reference 3 will be placed at position 1 and similarly \\2 and \\1 will be used.

In [ ]:

```

```

Question 20- Write a Python program to find all words starting with 'a' or 'e' in a given string.

In [30]:

```python
# Solution

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

list1 = re.findall("[ae]\w+", text)
print(list1)
```

```
['example', 'eates', 'an', 'ayList', 'apacity', 'elements', 'elements', 'are', 'en', 'ad
ded', 'ayList', 'and', 'ayList', 'ed', 'accordingly']
```

In [31]:

```python
#solution 2

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

pattern = "[ae][\w]+"
for match in re.finditer(pattern, text):
    s = match.start()
    e = match.end()
    print((text[s:e]))
```

```
example
eates
an
ayList
apacity
elements
elements
are
en
added
ayList
and
ayList
ed
accordingly
```

In [32]:

```python
#Solution 3

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

pattern = "[ae][\w]+"
for match in re.finditer(pattern, text):
    print(match.group(0))
```

```
example
eates
an
ayList
```

```
apacity
elements
elements
are
en
added
ayList
and
ayList
ed
accordingly
```

[ae][\w]+ - In this pattern [ae] is used to return the one or more occurence of character alphabetically between a and z, lower case or upper case and [\w]+ is used to return zero or more occurrence of any word characters (characters from a to Z, digits from 0-9, and the underscore _ character). So this pattern will return a string starting with 'a' or 'e' followed by any word.

In [ ]:

Question 21- Write a Python program to separate and print the numbers and their position of a given string.

In [33]:
```python
text = "The following example creates an ArrayList with a capacity of 50 elements. 4 ele

for m in re.finditer("\d+", text):
    print(m.group(0))
    print("Index position:", m.start())
```

```
50
Index position: 62
4
Index position: 75
```

In [ ]:

Question 22- Extract maximum numeric value from a string

In [34]:
```python
def extractMax(input):

    numbers = re.findall('\d+',input)
    print(numbers)

    numbers = map(int,numbers)        #converting each number from string into integer.

    print ("Maximum Numeric value is ",max(numbers))

input = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
extractMax(input)
```

```
['947', '896', '926', '524', '734', '950', '642']
Maximum Numeric value is  950
```

In [35]:
```python
def extractMax(input):
    numbers = re.findall('\d+',input)
    print(numbers)
    numbers = list(map(int,numbers))

    max = numbers[0]           #Assume first number in list is largest and initially we h
    for x in numbers:          #Traverse through the list
        if x > max:            #compare each number of the list with the value stored in
            max = x            #Whichever is largest assign that value to variable "max".
    return "Maximum Numeric value is", max                      #It will return the "max" valu
```

```
input = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
extractMax(input)
```

```
['947', '896', '926', '524', '734', '950', '642']
```

Out[35]:
```
('Maximum Numeric value is', 950)
```

In [ ]:

Question 23- Regex in Python to put spaces between words starting with capital letters

In [36]:
```
def putSpace(input):
    words = re.findall('[A-Z][a-z]*', input)
    print(' '.join(words))

input = 'BruceWayneIsBatman'
putSpace(input)
```

```
Bruce Wayne Is Batman
```

In this question we have used join() method to join all items in a list into a string, using a space character as separator

In [ ]:

Question 24- Python regex to find sequences of one upper case letter followed by lower case letters

In [37]:
```
def match(text):
        pattern = '[A-Z][a-z]*$'

        if re.search(pattern, text):
                return('Yes')
        else:
                return('No')


print(match("Welcome"))
print(match("WelcomeHomewelcome Hello Welcome"))
print(match("Welcomes you"))
```

```
Yes
Yes
No
```

In [ ]:

Question 25- Remove consecutive duplicate words from Sentence using Regular Expression

In [38]:
```
def removeDuplicateWords(input):

    regex = r'\b(\w+)(?:\W+\1\b)+'
    return re.sub(regex, r'\1', input)

# Test Case: 1
str1 = "Good bye bye world world"
print(removeDuplicateWords(str1))

# Test Case: 2
str2 = "Ram went went to to his home"
print(removeDuplicateWords(str2))

# Test Case: 3
```

```
str3 = "Hello hello world world"
print(removeDuplicateWords(str3))

# Test Case: 4
str4 = "Hello Hello world World"
print(removeDuplicateWords(str4))
```

```
Good bye world
Ram went to his home
Hello hello world
Hello world World
```

The details of the above regular expression can be understood as:

- “\b” - A word boundary. Boundaries are needed for special cases. For example, in “My thesis is great”, “is” wont be matched twice.
- “\w+” - A word character: [a-zA-Z_0-9]
- “\W+” - A non-word character: [^\w]
- “\1” - Matches whatever was matched in the 1st group of parentheses, which in this case is the (\w+)
- “+” - Match whatever it’s placed after 1 or more times

In [ ]:

Question 26- Program to accept string ending with alphanumeric character

In [39]:

```
regex = '[a-zA-z0-9]$'

def check_alpha_numeric(string):
    if(re.search(regex, string)):
        print("The string is ending with an alphanumeric character. \n")

    else:
        print("The string is not ending with an alphanumeric character. \n")


check_alpha_numeric("pitchumca@")
check_alpha_numeric("pitchumca123")
check_alpha_numeric("pitchum.")
check_alpha_numeric("staysafeindistancestay")
```

```
The string is not ending with an alphanumeric character.

The string is ending with an alphanumeric character.

The string is not ending with an alphanumeric character.

The string is ending with an alphanumeric character.
```

In [ ]:

Question 27- Write a python program using RegEx to extract the hashtags.

Sample Text: text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo"""

Output: [#Doltiwal', '#xyzabc', '#Demonetization]

In [40]:
```
text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the

hashtags = re.findall(r"#\w+", text)

print("Tweet:\n", text)
print("\n Hashtag:\n", hashtags)
```

```
Tweet:
 RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has
rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo

 Hashtag:
 ['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [41]:
```
#solution 2

text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the

hashtags = re.findall(r"#[a-zA-Z0-9_]+", text)     #

print("Tweet:\n", text)
print("\n Hashtag:\n", hashtags)
```

```
Tweet:
 RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has
rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo

 Hashtag:
 ['#Doltiwal', '#xyzabc', '#Demonetization']
```

hash[a-zA-Z0-9_]+-This pattern is used to match one or more characters that begins with a hash (#) and is followed by a lowercase letter or capital letter or a number or an underscore. A plus sign is used after square brackets to match one or more repetitions of the given pattern.

In [ ]:
```

```

Question 28- Write a python program using RegEx to remove <U+..> like symbols Check the below sample text, there are strange symbols something of the sort <U+..> all over the place. You need to come up with a general Regex expression that will cover/remove all such symbols.

Sample Text: "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

Output: @Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all different party leaders

In [42]:
```
text= "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who
clean_text = re.sub(r"<U\+[A-Z0-9]+>", "", text)

print("Text before:\n", text)
print("\n Text after:\n", clean_text)
```

```
Text before:
 @Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are pr
otesting #demonetization are all different party leaders

 Text after:
 @Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all
different party leaders
```

<U+[A-Z0-9]+>-This pattern is used to match one or more characters enclosed within <> that begin with 'U' followed by a plus sign and then a capital letter (upper case letters) or a number. A plus sign is used after square brackets to match one or more repetitions of a given pattern.

```
In [ ]:
```

Question 29- Write a python program to extract dates from the text stored in the text file.

Sample Text: Ron was born on 12-09-1992 and he was admitted to school 15-12-1999.

Store this sample text in the file and then extract dates.

```
In [43]:
import re
with open("C:/Users/LENOVO/Documents/RegEx/sample_text1.txt") as file:
        for line in file:
            emails = re.findall(r"\d{2}-\d{2}-\d{4}", line)
            print(emails)
```

```
['12-09-1992', '15-12-1999']
```

```
In [ ]:
```

Question 30- Write a Python program to replace all occurrences of a space, comma, or dot with a colon.

Sample Text- 'Python Exercises, PHP exercises.'

Output: Python:Exercises::PHP:exercises:

```
In [44]:
import re
text = 'Python Exercises, PHP exercises.'
print(re.sub("[ ,.]", ":", text))
```

```
Python:Exercises::PHP:exercises:
```

```
In [ ]:
```