REGULAR EXPRESSIONS

Q: 1 Write a Python program to replace all occurrences of a space , comma, or dot with a colon.

In [2]:
```python
text = 'Python Exercises, PHP exercises.'

# Replace space, comma, and dot with a colon
new_text = text.replace(' ' , ':').replace(',',':').replace('.',':')

print(new_text)
```
```
Python:Exercises::PHP:exercises:
```

Q:2 Create a dataframe using the dictionary below and remove everything(commas(,), !,XXXX,;,etc.)from the columns except words.

In [21]:
```python
import pandas as pd
import re

data = {'SUMMARY': ['hello,world!', 'XXXXX test', '123four,five:; six...']}
df = pd.DataFrame(data)
df['SUMMARY'] = df['SUMMARY'].str.replace('[^a-zA-Z\s]', '', regex=True)

print(df)
```
```
       SUMMARY
0    helloworld
1    XXXXX test
2  fourfive six
```

Q : 3 Create a function in python to find all words that are at least 4 charcters long in a string. The use of the re.compile()mehod is mandatory.

In [1]:
```python
import re

def find_words(string):
    pattern = re.compile(r'\b\w{3,5}\b')
    matches = pattern.findall(string)
    return matches
string = "That is a recompile method with words."
result = find_words(string)

print(result)
```
```
['That', 'with', 'words']
```

Q:4 Create a function in python to find all three ,four and five charcters words in a string. The use of the re.compile()method

In [2]:
```python
import re
text = 'The quick brown fox jumps over the lazy dog.'
print(re.findall(r"\b\w{3,5}\b", text))
```
```
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

Q:5 Create a function in Python to remove the parenthesis in a list of strings. The use of the re.compile() method is mandotory.

In [48]:
```python
import re

def remove_parenthese(strings):
    pattern = re.compile(r"\(\)")
    modified_strings = []
    modified_string = re.sub(pattern, "", string)
    modified_strings.aapend(modified_string)
    return modified_strings
    string = ["example(.com)","he@fliprobo(.com)","github(.com)","Hello(Data Science World)","Data(Scientist)"]
result = remove_parentheses(sample_text)
print(result)
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[48], line 10
      8     return modified_strings
      9     string = ["example(.com)","he@fliprobo(.com)","github(.com)","Hello(Data Science World)","Data(Scientist)"]
---> 10 result = remove_parentheses(sample_text)
     11 print(result)

NameError: name 'remove_parentheses' is not defined
```

Q: 6 Write a python program to remove the parenthesis area from the text stored in the text file using Regular Expression.

In [52]:
```python
import re

text = ["example(.com)","hr@fliprobo(.com)","github(.com)","Hello(Data Science World)","Dat(Scientist)"]

modified_text = re.sub(r'\([^()*\])', '', text)

print(modified_text)
```
```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Cell In[52], line 5
      1 import re
      3 text = ["example(.com)","hr@fliprobo(.com)","github(.com)","Hello(Data Science World)","Dat(Scientist)"]
----> 5 modified_text = re.sub(r'\([^()*\])', '', text)
      7 print(modified_text)

File D:\Users\HP\anaconda3\lib\re.py:209, in sub(pattern, repl, string, count, flags)
    202 def sub(pattern, repl, string, count=0, flags=0):
    203     """Return the string obtained by replacing the leftmost
    204     non-overlapping occurrences of the pattern in string by the
    205     replacement repl.  repl can be either a string or a callable;
    206     if a string, backslash escapes in it are processed.  If it is
    207     a callable, it's passed the Match object and must return
    208     a replacement string to be used."""
--> 209     return _compile(pattern, flags).sub(repl, string, count)

File D:\Users\HP\anaconda3\lib\re.py:303, in _compile(pattern, flags)
    301 if not sre_compile.isstring(pattern):
    302     raise TypeError("first argument must be string or compiled pattern")
--> 303 p = sre_compile.compile(pattern, flags)
    304 if not (flags & DEBUG):
    305     if len(_cache) >= _MAXCACHE:
    306         # Drop the oldest item

File D:\Users\HP\anaconda3\lib\sre_compile.py:788, in compile(p, flags)
    786 if isstring(p):
    787     pattern = p
--> 788     p = sre_parse.parse(p, flags)
    789 else:
    790     pattern = None

File D:\Users\HP\anaconda3\lib\sre_parse.py:955, in parse(str, flags, state)
    952 state.str = str
    954 try:
--> 955     p = _parse_sub(source, state, flags & SRE_FLAG_VERBOSE, 0)
    956 except Verbose:
    957     # the VERBOSE flag was switched on inside the pattern.  to be
    958     # on the safe side, we'll parse the whole thing again...
    959     state = State()

File D:\Users\HP\anaconda3\lib\sre_parse.py:444, in _parse_sub(source, state, verbose, nested)
    442 start = source.tell()
    443 while True:
--> 444     itemsappend(_parse(source, state, verbose, nested + 1,
    445                        not nested and not items))
    446     if not sourcematch("|"):
    447         break

File D:\Users\HP\anaconda3\lib\sre_parse.py:550, in _parse(source, state, verbose, nested, first)
    548 this = sourceget()
    549 if this is None:
--> 550     raise source.error("unterminated character set",
    551                        source.tell() - here)
    552 if this == "]" and set:
    553     break

error: unterminated character set at position 2
```

Q:7

In [2]:
```python
import re
text = "ImportanceOfRegularExpressionsInPython"

print(re.findall('[A-z][^A-Z]*',text))
```
```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

Q : 8

In [6]:
```python
import re

string="RegularExpression1IsAn2ImportantTopic3InPython"

words = re.findall('[A-Z][a-z]*', string)

print(' '.join((words)))
```
```
Regular Expression Is An Important Topic In Python
```

Q:9

In [55]:
```python
import re

string="RegularExpression1Is2ImportantTopic3InPython"

numbers = re.findall('[0-9]*', string)

print(' '.join((numbers)))
```
```
                 1   2                3
```

Q:10

In [ ]:

Q: 11

In [56]:
```python
import re
def text_match(text):
    patterns = '^[a-zA-Z0-9]*$'
    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("Python_Exercises_1"))
```
```
Not matched
Not matched
```

Q :12

In [57]:
```python
import re
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
```

Q : 13

In [15]:
```python
import re
ip = "29.08.094.196"
string = re.sub('\.[0]*', '.',ip)
print(string)
```
```
29.8.94.196
```

Q : 14

In [59]:
```python
import re

text = "On August 15th 1947 that India was declared independent from British colonialism,and the rein of control were handed over to the leaders of the country

patterns = r"\b([A-Z][a-z]+) \d{1,2}(?:st|nd|rd|th)? \d{4}\b"

matches = re.findall(pattern, text)
print(matches)
```
```
['On', '15th', '1947', 'that', 'was', 'from', 'and', 'the', 'rein', 'of', 'were', 'over', 'to', 'the', 'of', 'the']
```

Q : 15

In [60]:
```python
import re
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!')
    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!
```

Q : 16

In [61]:
```python
import re
pattern = 'fox'
text = 'The quick brown foxjumps 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 foxjumps over the lazy dog." from 16 to 19
```

Q:17

In [35]:
```python
import re
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"
```

Q:18

In [62]:
```python
import re
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
```

Q :19

In [18]:
```python
import re
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
```

Q:20

In [63]:
```python
import re

def find_decimal_numbers(string):
    pattern = re.compile(r'\d+\.\d{1,2}')
    sample_text = "01.120132.1232.31875145.8 3.01 27.25 0.25"
re.findall(pattern, string)
output = find_decimal_numbers(sample_text)

print(output)
```
```
None
```

Q:21

In [64]:
```python
import re
text = "The following example creates an ArrayList with a capacity of 50 elements."
for m in re.finditer("\d+", text):
    print(m.group(0))
    print("Index position:", m.start())
```
```
50
Index position: 62
```

Q:22

In [13]:
```python
import re
string = 'My marks in each sentence are :947,896,926,524,734,950,642'
number = re.findall('\d+', string)
number = map(int, number)
print("Max_value:",max(number))
```
```
Max_value: 950
```

Q:23

In [2]:
```python
import re
def capital_words_spaces(str1):
    return re.sub(r"(\w)([A-Z])", r"\1 \2", str1)

print(capital_words_spaces("RegularExpressionIsAnImportantTopicInPython"))
```
```
Regular Expression Is An Important Topic In Python
```

Q:24

In [65]:
```python
import re

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("AaBbCc"))
    print(text_match("Regular"))
    print(text_match("expression"))
```

Q :25

In [94]:
```python
import re

def remove_duplicates(sentence):
    pattern = r'\b(\w+)(\s+\1\b+'
    result = re.sub(pattern, r'\1', sentence)
    return result

sentence = "Hello hello world world"
result = remove_duplicates(sentence)

print(result)
```
```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Cell In[94], line 9
      6     return result
      8 sentence = "Hello hello world world"
----> 9 result = remove_duplicates(sentence)
     11 print(result)

Cell In[94], line 5, in remove_duplicates(sentence)
      3 def remove_duplicates(sentence):
      4     pattern = r'\b(\w+)(\s+\1\b+'
----> 5     result = re.sub(pattern, r'\1', sentence)
      6     return result

File D:\Users\HP\anaconda3\lib\re.py:209, in sub(pattern, repl, string, count, flags)
    202 def sub(pattern, repl, string, count=0, flags=0):
    203     """Return the string obtained by replacing the leftmost
    204     non-overlapping occurrences of the pattern in string by the
    205     replacement repl.  repl can be either a string or a callable;
    206     if a string, backslash escapes in it are processed.  If it is
    207     a callable, it's passed the Match object and must return
    208     a replacement string to be used."""
--> 209     return _compile(pattern, flags).sub(repl, string, count)

File D:\Users\HP\anaconda3\lib\re.py:303, in _compile(pattern, flags)
    301 if not sre_compile.isstring(pattern):
    302     raise TypeError("first argument must be string or compiled pattern")
--> 303 p = sre_compile.compile(pattern, flags)
    304 if not (flags & DEBUG):
    305     if len(_cache) >= _MAXCACHE:
    306         # Drop the oldest item

File D:\Users\HP\anaconda3\lib\sre_compile.py:788, in compile(p, flags)
    786 if isstring(p):
    787     pattern = p
--> 788     p = sre_parse.parse(p, flags)
    789 else:
    790     pattern = None

File D:\Users\HP\anaconda3\lib\sre_parse.py:955, in parse(str, flags, state)
    952 state.str = str
    954 try:
--> 955     p = _parse_sub(source, state, flags & SRE_FLAG_VERBOSE, 0)
    956 except Verbose:
    957     # the VERBOSE flag was switched on inside the pattern.  to be
    958     # on the safe side, we'll parse the whole thing again...
    959     state = State()

File D:\Users\HP\anaconda3\lib\sre_parse.py:444, in _parse_sub(source, state, verbose, nested)
    442 start = source.tell()
    443 while True:
--> 444     itemsappend(_parse(source, state, verbose, nested + 1,
    445                        not nested and not items))
    446     if not sourcematch("|"):
    447         break

File D:\Users\HP\anaconda3\lib\sre_parse.py:841, in _parse(source, state, verbose, nested, first)
    838         raise source.error(err.msg, len(name) + 1) from None
    839 sub_verbose = ((verbose or (add_flags & SRE_FLAG_VERBOSE)) and
    840                not (del_flags & SRE_FLAG_VERBOSE))
--> 841 p = _parse_sub(source, state, sub_verbose, nested + 1)
    842 if not source.match(")"):
    843     raise source.error("missing ), unterminated subpattern",
    844                        source.tell() - start)

File D:\Users\HP\anaconda3\lib\sre_parse.py:444, in _parse_sub(source, state, verbose, nested)
    442 start = source.tell()
    443 while True:
--> 444     itemsappend(_parse(source, state, verbose, nested + 1,
    445                        not nested and not items))
    446     if not sourcematch("|"):
    447         break

File D:\Users\HP\anaconda3\lib\sre_parse.py:669, in _parse(source, state, verbose, nested, first)
    667     item = None
    668 if not item or item[0][0] is AT:
--> 669     raise source.error("nothing to repeat",
    670                        source.tell() - here + len(this))
    671 if item[0][0] in _REPEATCODES:
    672     raise source.error("multiple repeat",
    673                        source.tell() - here + len(this))

error: nothing to repeat at position 15
```

In [ ]:
```
Q: 26
```

In [72]:
```python
import re

regex_expression = '[a-zA-Z0-9]$'

def check_string(my_string):

    if(re.search(regex_expression, my_string)):
        print("The string ends with alphanumeric character")

    else:
        print("The string doesnot end with alphanumeric character")


my_string_1 = "Python@"
print("The string is :")
print(my_string_1)
check_string(my_string_1)

my_string_2 = "Python12345"
print("\nThe string is :")
print(my_string_2)
check_string(my_string_2)
```
```
The string is :
Python@
The string doesnot end with alphanumeric character

The string is :
Python12345
The string ends with alphanumeric character
```

Q:27

In [28]:
```python
import re

def extract_hashtags(text):
    hashtags = re.findall(r'#\w+', text)
    return hashtags
text = """RT@kapil_kausik:#DotiwalImean#xyzabcis"hurt"by#Demonetizattion as the same has rendered USELESS<ed><U+00B1><U+0089>"acquiredfunds"No wo"""
hashtags = extract_hashtags(text)
print(hashtags)
```
```
['#DotiwalImean', '#xyzabcis', '#Demonetizattion']
```

Q:28

In [31]:
```python
import re

input_text = "@Jags123456Bharat band on 28??<ed><U+00A0><U+00B8><U+0082>Those who are protesting#demonetization are all different party leaders"

pattern = r"<U\+\w{4}>"
output_text = re.sub(pattern, "", input_text)

print(output_text)
```
```
@Jags123456Bharat band on 28??<ed>Those who are protesting#demonetization are all different party leaders
```

Q:29

In [38]:
```python
import re

text = "Ron was born on 12-9-1992 and he was admitted to school 15-12-1999."

pattern = r'\d{2}-\d{2}-\d{4}'
dates = re.findall(pattern, text)

print(dates)
```
```
['15-12-1999']
```

Q:30

In [40]:
```python
import re

sample_text = "The following example creates an Arraylist with a capacity of 50 elements. 4 elements are then added to the Arraylist is trimmed accordingly."

pattern = re.compile(r'\b\w{2,4}\b')
modified_string = re.sub(pattern, '', sample_text)

print(modified_string)
```
```
 following example creates  Arraylist  a capacity   elements. 4 elements   added   Arraylist  trimmed accordingly.
```