In [1]:

```
def replace_chars(text):
    replacements = {' ': ':', ',': ':', '.': ':'}
    for char, replacement in replacements.items():
        text = text.replace(char, replacement)
    return text

sample_text = 'Python Exercises, PHP exercises.'
result = replace_chars(sample_text)
print(result)
```

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

In [2]:

```
 import pandas as pd
import re

data = {'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:;
six...']}
df = pd.DataFrame(data)

def remove_non_words(text):
    return re.sub(r'[^\w\s]', '', text)

df['SUMMARY'] = df['SUMMARY'].apply(remove_non_words)
```

```
print(df)
[10:07 PM, 2/6/2024] shubh Shekhar: def replace_characters(text):
    replacements = {' ': ':', ',': ':', '.': ':'}
    for old_char, new_char in replacements.items():
        text = text.replace(old_char, new_char)
    return text

sample_text = 'Python Exercises, PHP exercises.'
output = replace_characters(sample_text)
print(output)
[10:19 PM, 2/6/2024] shubh Shekhar: import pandas as pd
import re

data = {'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:;
six...']}
df = pd.DataFrame(data)

def clean_text(text):
    # Use regex to remove unwanted characters
    cleaned_text = re.sub(r'[^\w\s]', '', text)
    return cleaned_text

# Apply clean_text function to each element in the 'SUMMARY' column
df['SUMMARY'] = df['SUMMARY'].apply(clean_text)

print(df)
```

```
 Cell In[2], line 13
   [10:07 PM, 2/6/2024] shubh Shekhar: def replace_characters(text):
      ^
SyntaxError: leading zeros in decimal integer literals are not permitted; use
an 0o prefix for octal integers
```

In [1]:

```
import re

def find_long_words(text):
    pattern = re.compile(r'\b\w{4,}\b')
    return pattern.findall(text)

# Example usage:
text = "This is a sample sentence with words of varying lengths like apple,
banana, and orange."
long_words = find_long_words(text)
print(long_words)
```

```
['This', 'sample', 'sentence', 'with', 'words', 'varying', 'lengths', 'like',
'apple', 'banana', 'orange']
```

In [2]:

```
import re

def find_specific_length_words(text):
    pattern = re.compile(r'\b\w{3,5}\b')
    return pattern.findall(text)

# Example usage:
text = "This is a sample sentence with words of varying lengths like apple,
banana, and orange."
specific_length_words = find_specific_length_words(text)
print(specific_length_words)
```

```
['This', 'with', 'words', 'like', 'apple', 'and']
```

In [3]:

```
import re

def remove_parentheses(strings):
    pattern = re.compile(r'\(([^)]+)\)')
    return [pattern.sub('', s) for s in strings]

# Example usage:
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)",
"Hello (Data Science World)", "Data (Scientist)"]
output = remove_parentheses(sample_text)
for string in output:
    print(string)
```

```
example
hr@fliprobo
github
Hello
Data
```

In [4]:

```
import re
```

```python
# Read text from file
with open('sample_text.txt', 'r') as file:
    text = file.read()

# Remove parenthesis area using regular expression
modified_text = re.sub(r'\s*\([^)]*\)', '', text)

# Print modified text
print(modified_text)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[4], line 4
      1 import re
      3 # Read text from file
----> 4 with open('sample_text.txt', 'r') as file:
      5     text = file.read()
      7 # Remove parenthesis area using regular expression

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:284, in
_modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you are
doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'sample_text.txt'
```

In [5]:

```python
import re

# Sample text
sample_text = "ImportanceOfRegularExpressionsInPython"

# Split the string into uppercase letters
result = re.findall('[A-Z][^A-Z]*', sample_text)

# Print the result
print(result)
```

```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [6]:

```python
import re

# Read text from the file
with open('sample_text.txt', 'r') as file:
    text = file.read()

# Define pattern to match text within parentheses
pattern = r'\s*\([^)]*\)'

# Remove text within parentheses using regular expression
cleaned_text = re.sub(pattern, '', text)

# Convert cleaned text to list
cleaned_text_list = cleaned_text.split(',')

# Remove leading and trailing whitespaces from each element in the list
cleaned_text_list = [text.strip() for text in cleaned_text_list]
```

```
print(cleaned_text_list)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[6], line 4
      1 import re
      3 # Read text from the file
----> 4 with open('sample_text.txt', 'r') as file:
      5     text = file.read()
      7 # Define pattern to match text within parentheses

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:284, in
_modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you are
doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'sample_text.txt'
```

In [7]:

```
import re

text = "ImportanceOfRegularExpressionsInPython"
result = re.findall('[A-Z][^A-Z]*', text)
print(result)
```

```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [8]:

```
import re

def insert_spaces(text):
    # Use regular expression to find words starting with numbers
    pattern = r'\b(?=\d)'
    matches = re.finditer(pattern, text)

    # Iterate through matches and insert spaces before the numbers
    offset = 0
    for match in matches:
        start_index = match.start() + offset
        text = text[:start_index] + ' ' + text[start_index:]
        offset += 1  # Increment offset to account for inserted space

    return text

# Sample Text
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Insert spaces
output_text = insert_spaces(sample_text)
print(output_text)
```

```
RegularExpression1IsAn2ImportantTopic3InPython
```

In [9]:

```
import re

def insert_spaces(text):
    # Using regular expression to find words starting with capital letters
or numbers
    pattern = r'(?<=[a-z])(?=[A-Z0-9])|\d(?=\D)'
    # Inserting spaces between words starting with capital letters or
numbers
    spaced_text = re.sub(pattern, ' ', text)
    return spaced_text

# Example usage:
sample_text = "RegularExpression1IsAn2ImportantTopic3InPyt"
spaced_text = insert_spaces(sample_text)
print(spaced_text)
```

```
Regular Expression  Is An  Important Topic  In Pyt
```

In [ ]:

```
import pandas as pd

# Read data from GitHub link into a dataframe
url =
"https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_sco
re_dataset.csv"
df = pd.read_csv(url)
```

```
# Extract first 6 letters of each country and store in a new column called
"first_five_letters"
df['first_five_letters'] = df['Country'].str[:6]

# Display the dataframe
print(df.head())
```

In [11]:

```
import re

def match_string(string):
    # Regular expression pattern to match the criteria
    pattern = r'^[a-zA-Z0-9_]+$'

    # Check if the string matches the pattern
    if re.match(pattern, string):
        return True
    else:
        return False

# Test the function
test_strings = ["Hello_World123", "hello123", "123", "hello world",
"hello@world"]
for string in test_strings:
    if match_string(string):
        print(f'"{string}" matches the criteria.')
    else:
        print(f'"{string}" does not match the criteria.')
```

```
"Hello_World123" matches the criteria.
"hello123" matches the criteria.
"123" matches the criteria.
```

```
"hello world" does not match the criteria.
"hello@world" does not match the criteria.
```

In [12]:

```
import re

def starts_with_number(string, number):
    # Regular expression pattern to match the specific number at the
beginning of the string
    pattern = r'^' + str(number) + r'\D'

    # Check if the string matches the pattern
    if re.match(pattern, string):
        return True
    else:
        return False

# Test the function
test_strings = ["123Hello", "456World", "789Goodbye", "Hello123",
"world456"]
specific_number = 123
for string in test_strings:
    if starts_with_number(string, specific_number):
        print(f'"{string}" starts with the specific number
{specific_number}.')
    else:
        print(f'"{string}" does not start with the specific number
{specific_number}.')
```

```
"123Hello" starts with the specific number 123.
"456World" does not start with the specific number 123.
"789Goodbye" does not start with the specific number 123.
```

```
"Hello123" does not start with the specific number 123.
"world456" does not start with the specific number 123.
```

In [13]:

```
import re

def remove_leading_zeros(ip_address):
    # Regular expression pattern to remove leading zeros
    pattern = r'\b0+(\d+)\b'

    # Replace leading zeros with the non-zero digits
    modified_ip = re.sub(pattern, r'\1', ip_address)

    return modified_ip

# Test the function
ip_address = "192.168.001.001"
modified_ip = remove_leading_zeros(ip_address)
print("Original IP address:", ip_address)
print("Modified IP address:", modified_ip)
```

```
Original IP address: 192.168.001.001
Modified IP address: 192.168.1.1
```

In [14]:

```
import re
```

```
# Sample text
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."

# Regular expression pattern to match date string
pattern =
r'\b(?:January|February|March|April|May|June|July|August|September|October|
November|December)\s+\d{1,2}(?:st|nd|rd|th)?\s+\d{4}\b'

# Find all matches
matches = re.findall(pattern, text)

# Print matches
print(matches)
```

```
['August 15th 1947']
```

In [15]:

```
def search_literals(text, literals):
    found_indices = []
    for literal in literals:
        index = text.find(literal)
        if index != -1:
            found_indices.append((literal, index))

    return found_indices

# Sample text
sample_text = 'The quick brown fox jumps over the lazy dog.'
```

```
# List of literals to search for
literals_to_search = ['brown', 'fox', 'cat', 'dog']

# Using find() method
found_indices = search_literals(sample_text, literals_to_search)
if found_indices:
    print("Using find() method:")
    for literal, index in found_indices:
        print(f"'{literal}' found at index {index}")
else:
    print("No literals found using find() method.")

# Using 'in' operator
print("\nUsing 'in' operator:")
for literal in literals_to_search:
    if literal in sample_text:
        print(f"'{literal}' found.")
    else:
        print(f"'{literal}' not found.")
```

```
Using find() method:
'brown' found at index 10
'fox' found at index 16
'dog' found at index 40

Using 'in' operator:
'brown' found.
'fox' found.
'cat' not found.
'dog' found.
```

In [16]:

```python
def find_substrings(text, substring):
    indices = []
    start_index = 0
    while True:
        index = text.find(substring, start_index)
        if index == -1:
            break
        indices.append(index)
        start_index = index + 1
    return indices

# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'

# Substring to find
substring_to_find = 'exercises'

# Find the substrings
substrings_indices = find_substrings(sample_text, substring_to_find)

# Print the result
if substrings_indices:
    print(f"'{substring_to_find}' found at indices:", substrings_indices)
else:
    print(f"'{substring_to_find}' not found in the sample text.")
```

```
'exercises' found at indices: [7, 22, 36]
```

In [17]:

```python
def find_substring_occurrences(text, substring):
    occurrences = []
    start_index = 0
    while True:
        index = text.find(substring, start_index)
        if index == -1:
            break
        occurrences.append((substring, index))
        start_index = index + 1
    return occurrences

# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'

# Substrings to find
substrings_to_find = ['exercises', 'Python', 'PHP', 'C#']

# Find the occurrences of substrings
all_occurrences = []
for substring in substrings_to_find:
    substring_occurrences = find_substring_occurrences(sample_text,
substring)
    all_occurrences.extend(substring_occurrences)

# Print the result
if all_occurrences:
    print("Occurrences and positions of substrings:")
    for occurrence in all_occurrences:
        substring, position = occurrence
        print(f"'{substring}' found at position {position}")
else:
    print("No occurrences found.")
```

```
Occurrences and positions of substrings:
'exercises' found at position 7
'exercises' found at position 22
'exercises' found at position 36
'Python' found at position 0
'PHP' found at position 18
'C#' found at position 33
```

In [18]:

```python
from datetime import datetime

def convert_date_format(date_str):
    # Convert string to datetime object
    date_obj = datetime.strptime(date_str, '%Y-%m-%d')
    # Convert datetime object to string in desired format
    new_date_str = date_obj.strftime('%d-%m-%Y')
    return new_date_str

# Sample date string in yyyy-mm-dd format
date_str = '2024-02-07'

# Convert date format
new_date_str = convert_date_format(date_str)

# Print the result
print("Original date:", date_str)
print("Converted date:", new_date_str)
```

```
Original date: 2024-02-07
Converted date: 07-02-2024
```

In [19]:

```python
import re

def find_decimal_numbers(text):
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')
    matches = pattern.findall(text)
    return matches

# Sample text
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

# Find all decimal numbers with precision of 1 or 2
decimal_numbers = find_decimal_numbers(sample_text)

# Print the result
print("Decimal numbers with precision of 1 or 2:")
print(decimal_numbers)
```

```
Decimal numbers with precision of 1 or 2:
['01.12', '145.8', '3.01', '27.25', '0.25']
```

In [20]:

```
import re

def find_numbers_with_positions(text):
    pattern = re.compile(r'\b\d+\b')
    matches = pattern.finditer(text)
    numbers_with_positions = [(match.group(), match.start()) for match in
matches]
    return numbers_with_positions

# Sample text
sample_text = "There are 10 apples and 20 oranges on the table."

# Find numbers and their positions
numbers_with_positions = find_numbers_with_positions(sample_text)
```

```
# Print the result
print("Numbers and their positions:")
for number, position in numbers_with_positions:
    print(f"Number: {number}, Position: {position}")
```

```
Numbers and their positions:
Number: 10, Position: 10
Number: 20, Position: 24
```

In [21]:

```
import re

def extract_maximum_number(text):
    numbers = re.findall(r'\b\d+\b', text)
    if numbers:
        max_number = max(map(int, numbers))
        return max_number
    else:
        return None

# Sample text
sample_text = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 
642'

# Extract maximum number
max_number = extract_maximum_number(sample_text)

# Print the result
if max_number is not None:
    print("Maximum number:", max_number)
else:
    print("No numbers found in the sample text.")
```

```
Maximum number: 950
```

In [22]:

```
import re

def insert_spaces(text):
    # Using regular expression to find words starting with capital letters
    pattern = r'(?<=[a-z])(?=[A-Z])'
    # Inserting spaces between words starting with capital letters
    spaced_text = re.sub(pattern, ' ', text)
    return spaced_text

# Sample text
sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces between words starting with capital letters
spaced_text = insert_spaces(sample_text)

# Print the result
print(spaced_text)
```

```
Regular Expression Is An Important Topic In Python
```

In [23]:

```
import re

# Sample text
sample_text = "Hello World, OpenAI is an Amazing Tool."

# Regular expression pattern
pattern = r'[A-Z][a-z]+'

# Find all matches
matches = re.findall(pattern, sample_text)

# Print the matches
print(matches)
```

```
['Hello', 'World', 'Open', 'Amazing', 'Tool']
```

In [24]:

```
import re

def remove_continuous_duplicates(sentence):
    # Regular expression pattern to match continuous duplicate words
    pattern = r'\b(\w+)(\s+\1)+\b'

    # Replace continuous duplicate words with a single occurrence
    cleaned_sentence = re.sub(pattern, r'\1', sentence)

    return cleaned_sentence
```

```
# Sample text
sample_text = "Hello hello world world"

# Remove continuous duplicate words
cleaned_text = remove_continuous_duplicates(sample_text)

# Print the result
print(cleaned_text)
```

```
Hello hello world
```

In [25]:

```
import re

def ends_with_alphanumeric(text):
    # Regular expression pattern to match string ending with an
alphanumeric character
    pattern = r'\w$'

    # Check if the string matches the pattern
    if re.search(pattern, text):
        return True
    else:
        return False

# Test the function
test_strings = ["hello123", "world_", "12345", "end$"]
for string in test_strings:
    if ends_with_alphanumeric(string):
        print(f'"{string}" ends with an alphanumeric character.')
    else:
        print(f'"{string}" does not end with an alphanumeric character.')
```

```
"hello123" ends with an alphanumeric character.
"world_" ends with an alphanumeric character.
"12345" ends with an alphanumeric character.
"end$" does not end with an alphanumeric character.
```

In [26]:

```python
import re

def extract_hashtags(text):
    # Regular expression pattern to match hashtags
    pattern = r'#\w+'

    # Find all matches
    hashtags = re.findall(pattern, text)

    return hashtags

# Sample text
sample_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"""

# Extract hashtags
hashtags = extract_hashtags(sample_text)

# Print the result
print("Extracted hashtags:")
print(hashtags)
```

```
Extracted hashtags:
['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [27]:

```python
import re

def extract_hashtags(text):
    # Regular expression pattern to match hashtags
    pattern = r'#\w+'

    # Find all matches
    hashtags = re.findall(pattern, text)

    return hashtags

# Sample text
sample_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"""

# Extract hashtags
hashtags = extract_hashtags(sample_text)

# Print the result
print("Extracted hashtags:")
print(hashtags)
```

```
Extracted hashtags:
['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [28]:

```python
import re

def remove_unicode_symbols(text):
    # Regular expression pattern to match <U+..> like symbols
    pattern = r'<U\+[A-Fa-f0-9]+>'

    # Replace the symbols with an empty string
    cleaned_text = re.sub(pattern, '', text)

    return cleaned_text

# Sample text
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"

# Remove <U+..> like symbols
cleaned_text = remove_unicode_symbols(sample_text)

# Print the result
print("Cleaned text:")
print(cleaned_text)
```

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

In [29]:

```
import re

def remove_unicode_symbols(text):
    # Regular expression pattern to match <U+..> like symbols
    pattern = r'<U\+[A-Fa-f0-9]+>'

    # Replace the symbols with an empty string
    cleaned_text = re.sub(pattern, '', text)

    return cleaned_text

# Sample text
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"

# Remove <U+..> like symbols
cleaned_text = remove_unicode_symbols(sample_text)

# Print the result
print("Cleaned text:")
print(cleaned_text)
```

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

In [30]:

```
import re
```

```python
def extract_dates_from_text(file_path):
    dates = []
    # Open the file and read its contents
    with open(file_path, 'r') as file:
        text = file.read()
        # Regular expression pattern to match dates in the format
DD-MM-YYYY
        pattern = r'\b\d{2}-\d{2}-\d{4}\b'
        # Find all matches
        dates = re.findall(pattern, text)
    return dates

# Sample text file path
file_path = "sample_text.txt"

# Extract dates from the text file
dates = extract_dates_from_text(file_path)

# Print the result
print("Extracted dates:")
print(dates)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[30], line 18
     15 file_path = "sample_text.txt"
     17 # Extract dates from the text file
---> 18 dates = extract_dates_from_text(file_path)
     20 # Print the result
     21 print("Extracted dates:")

Cell In[30], line 6, in extract_dates_from_text(file_path)
      4 dates = []
      5 # Open the file and read its contents
----> 6 with open(file_path, 'r') as file:
      7     text = file.read()
      8     # Regular expression pattern to match dates in the format
DD-MM-YYYY
```

```
File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:284, in
_modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you are
doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'sample_text.txt'
```

In [31]:

```python
import re

def remove_words_between_length_2_and_4(text):
    # Regular expression pattern to match words of length between 2 and 4
    pattern = re.compile(r'\b\w{2,4}\b')
    # Remove words matching the pattern
    cleaned_text = pattern.sub('', text)
    return cleaned_text

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

# Remove words of length between 2 and 4
cleaned_text = remove_words_between_length_2_and_4(sample_text)

# Print the result
print(cleaned_text)
```

following example creates ArrayList a capacity elements. 4 elements added ArrayList ArrayList trimmed accordingly.

In [ ]: