regularexpresion

July 16, 2023

[1]:
```
import re
```

[2]:
```
def check_string(input_string):
    pattern = r'^[a-zA-Z0-9]+$'
    match = re.match(pattern, input_string)
    return match is not None
```

[3]:
```
input_str = input("Enter a string: ")
if check_string(input_str):
    print("The string contains only the specified characters.")
else:
    print("The string contains characters other than a-z, A-Z, and 0-9.")
```

```
Enter a string: a-z, A-Z and 0-9
The string contains characters other than a-z, A-Z, and 0-9.
```

[4]:
```
#Create a function in python that matches a string that has an a followed by␣
 ↪zero or more b's
```

[5]:
```
def match_pattern(input_string):
    pattern = r'ab*'
    match = re.match(pattern, input_string)
    return match is not done
```

[ ]:
```
input_str = input("Enter a string: ")
if match_pattern(input_str):
    print("The string matches the patter: 'a' followed by zero or more 'b's.")
else:
    print("The String does not match the pattern.")
```

[ ]:
```
#Create a function in python that matches a string that has an a followed by␣
 ↪one or more b's
```

[9]:
```
import re

def match_string(input_string):
    patter = r'a+b+'
    match = re.match(pattern, input_string)
```

```
    if match:
        retrun True
    else:
        return False
```

```
  File "C:\Users\ADMIN\AppData\Local\Temp\ipykernel_10400\2636506139.py", line 
    retrun True
           ^
SyntaxError: invalid syntax
```

[ ]:
```
print(match_string('ab'))
print(match_string('abb'))
print(match_string('aab'))
print(match_string('abc'))
print(match_string('axbbbb'))
```

[ ]:
```
#Write a Python program that matches a string that has an a followed by three␣
↪'b'.
```

[ ]:
```
print(match_string('abbb'))
print(match_string('abb'))
print(match_string('aabbbb'))
print(match_string('abbbbb'))
print(match_string('axbbbbb'))
```

[5]:
```
#Write a Python program that matches a string that has an a followed by three␣
↪'b'.
```

[6]:
```
import re

def match_string(string):
    pattern = r'a{1}b{3}'
    if re.match(pattern, string):
        print("Match found!")
    else:
        print("No match found.")
```

[7]:
```
match_string("abbb")
```

```
Match found!
```

[10]:
```
#ANSWER FOR QUESTION 6
```

[12]:
```
import re

text = "IportanceOfRegularExpressionsInPython"
```

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

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

[13]:
```
#ANSWER FOR QUESTION 7
```

[14]:
```
import re

def match_string(string):
    pattern = r'a{1}b{2,3}'
    if re.match(pattern, string):
        print("Match Found!")
    else:
        print("No match found.")
```

[15]:
```
match_string("ab")
```

```
No match found.
```

[16]:
```
match_string("abbb")
```

```
Match Found!
```

[17]:
```
match_string("abbbb")
```

```
Match Found!
```

[18]:
```
#ANSWER FOR QUESTION 8
```

[19]:
```
import re

def find_sequences(string):
    pattern = r'[a-z]+_[a-z]+'
    sequences = re.findall(pattern, string)
    return sequences
```

[20]:
```
text = "hello_world, this_is_a_sequence, python_programming"
sequences = find_sequences(text)
print(sequences)
```

```
['hello_world', 'this_is', 'a_sequence', 'python_programming']
```

[21]:
```
#ANSWER FOR QUESTION 9
```

[23]:
```
import re

def match_string(string):
    pattern = r'a.*b$'
    if re.match(pattern, string):
```

```
        print("Match found!")
    else:
        print("No match found.")
```

[24]:
```
match_string("ab")
```

```
Match found!
```

[25]:
```
match_string("abc")
```

```
No match found.
```

[26]:
```
#ANSWER FOR QUESTION 10
```

[27]:
```
import re

def match_word(string, word):
    pattern = r'^' + re.escape(word)
    if re.match(pattern, string):
        print("Match found!")
    else:
        print("No match found.")
```

[28]:
```
match_word("Hello world", "Hello")
```

```
Match found!
```

[29]:
```
match_word("Hello world", "world")
```

```
No match found.
```

[30]:
```
#ANSWER FOR QUESTION 11
```

[5]:
```
import re

def is_valid_string(input_string):
    pattern = r'^[a-zA-Z0-9_]+$'
    match = re.match(pattern, input_string)
    return bool(match)
```

[6]:
```
strings = ['Hello123', 'ABC_', '12_34', '@#$%', 'abc123!', 'Python_Code']
for string in strings:
    if is_valid_string(string):
        print(f'{string}: Valid')
    else:
        print(f'{string}: Invalid')
```

```
Hello123: Valid
ABC_: Valid
12_34: Valid
```

```
@#$%: Invalid
abc123!: Invalid
Python_Code: Valid
```

[7]:
```python
#Write a Python program where a string will start with a specific number.
```

[11]:
```python
def starts_with_number(input_string, number):
    if input_string.startswith(str(number)):
        return True
    return False
```

[12]:
```python
strings = ['123abc', '456def', '789xyz', 'abc123', 'def456']
specific_number = 123
for string in strings:
    if starts_with_number(string, specific_number):
        print(f'{string}: Starts with {specific_number}')
    else:
        print(f'{string}: Starts with {specific_number}')
```

```
123abc: Starts with 123
456def: Starts with 123
789xyz: Starts with 123
abc123: Starts with 123
def456: Starts with 123
```

[13]:
```python
#Write a Python program to remove leading zeros from an IP address
```

[19]:
```python
def reomve_leading_zeros(ip_address):
    components = [str(int(component)) for component in components]
    cleaned_ip = '.'.join(components)
    return cleaned_ip
ip_addresses = ['192.168.001.001', '010.010.010.010', '000.001.002.003', '127.
 ↪000.000.001']
for ip in ip_addresses:
    cleaned_ip = remove_leading_zeros(ip)
    print(f'Original IP: {ip}\tCleaned IP: {cleaned_ip}')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2224\1932388896.py in <module>
      5 ip_addresses = ['192.168.001.001', '010.010.010.010', '000.001.002.003'
 ↪'127.000.000.001']
      6 for ip in ip_addresses:
----> 7     cleaned_ip = remove_leading_zeros(ip)
      8     print(f'Original IP: {ip}\tCleaned IP: {cleaned_ip}')

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

[20]:
```
#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.
```

[28]:
```
import re

def extract_date_from_text(file_path):
    with open(file_path, 'r')as file:
        text = file.read()
        pattern = r'\b([A-Z][a-z]+ \d{1,2}(?:st|nd|rd|th)? \d{4})\b'
        match = re.search(pattern, text)
        if match:
            return match.group(1)
        else:
            return None
```

[32]:
```
file_path = r'C:\Users\ADMIN\OneDrive\Documents\my_text.txt
date = extract_date_from_text(file_path)
if date:
    print(f"Date found: {date}")
else:
    print("No date found in the text.")
```

```
  File "C:\Users\ADMIN\AppData\Local\Temp\ipykernel_2224\2407166445.py", line 1
    file_path = r'C:\Users\ADMIN\OneDrive\Documents\my_text.txt
                                                              ^
SyntaxError: EOL while scanning string literal
```

[33]:
```
#Write a Python program to search some literals strings in a string
```

[34]:
```
def search_literals(text, words):
    found_words = []
    for word in words:
        if word in text:
            found_words.append(word)
    return found_words
```

[36]:
```
sample_text = 'The quick brown fox jumps over  the lazy dog.'
searched_words = ['fox', 'dog', 'horses']

found_words = search_literals(sample_text, searched_words)

print(f"Text: {sample_text}")
print("Found words:", found_words)
```

```
Text: The quick brown fox jumps over  the lazy dog.
Found words: ['fox', 'dog']
```

[37]:
```
#Write a Python program to find the substrings within a string.
```

[38]:
```
import re

def find_substrings(text, pattern):
    matches = re.findall(pattern, text)
    return matches

sample_text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'excercise'

substrings = find_substrings(sample_text, pattern)

print(f"Text: {sample_text}")
print(f"Pattern: {pattern}")
print("Substrings:")
for substring in substrings:
    print(substring)
```

```
Text: Python exercises, PHP exercises, C# exercises
Pattern: excercise
Substrings:
```

[39]:
```
#ANSWER FOR QUESTION 18
```

[45]:
```
def find_substrings_occurrences(text, substring):
    occurrences = []
    start = 0
    while True:
        index = text.find(substring, start)
        if index == -1:
            break
        occurrences.append((substring, index))
        start = index + 1
    return occurrences

sample_text = 'Python exercises, PHP exercises, C# exercises'
substrings = ['exercises', 'PHP', 'Java']

for substring in substrings:
    occurrences = find_substring_occurrences(sample_text, substring)
    print(f"Substring: {substring}")
    if occurrences:
        print("Occurrences:")
        for occurrence in occurrences:
            substring, position = occurrence
            print(f"Position: {position}\tSubstring: {substring}")
        else:
```

```
            print("No occurrences found.")
        print()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2224\4275409902.py in <module>
     14
     15 for substring in substrings:
---> 16     occurrences = find_substring_occurrences(sample_text, substring)
     17     print(f"Substring: {substring}")
     18     if occurrences:

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

[46]:
```
#ANSWER FOR QUETION 19
```

[47]:
```
from datetime import datetime

def convert_date_format(date_str):
    date_obj = datetime.strptime(date_str, '%Y-%m-%d')
    formatted_date = date_obj.strftime('%d-%m-%Y')
    return formatted_date
```

[48]:
```
date_str = '2023-07-16'
formatted_date = convert_date_format(date_str)
print(f"Original date: {date_str}")
print(f"Formatted date: {formatted_date}")
```

```
Original date: 2023-07-16
Formatted date: 16-07-2023
```

[49]:
```
#ANSWER FOR QUESTION 20
```

[51]:
```
import re

def find_words_starting_with_a_or_e(input_string):
    pattern = r'\b[ae]\w+\b'
    matches = re.findall(pattern, input_string, re.IGNORECASE)
    return matches
```

[52]:
```
sample_string = 'An apple and an elephant entered the elevator.'
words = find_words_starting_with_a_or_e(sample_string)
print(f"Input string: {sample_string}")
print(f"Input string: {sample_string}")
print(f"Words starting with 'a' or 'e': {words}")
```

```
Input string: An apple and an elephant entered the elevator.
Input string: An apple and an elephant entered the elevator.
Words starting with 'a' or 'e': ['An', 'apple', 'and', 'an', 'elephant',
'entered', 'elevator']
```

[53]:
```
#ANSWER FOR QUESTION 21
```

[54]:
```
import re

def separate_numbers_and_positions(input_string):
    pattern = r'\b\d+\b'
    matches = re.finditer(pattern, input_string)
    numbers_and_positions = [(match.group(), match.start()) for match in␣
 ↪matches]
    return numbers_and_positions
```

[55]:
```
sample_string = 'The price is $10.50 and the quantity is 25.'
result = separate_numbers_and_positions(sample_string)
print(f"Input string: {sample_string}")
print("Numbers and their positions:")
for number, position in result:
    print(f"Number: {number}\tPosition: {position}")
```

```
Input string: The price is $10.50 and the quantity is 25.
Numbers and their positions:
Number: 10      Position: 14
Number: 50      Position: 17
Number: 25      Position: 40
```

[56]:
```
#ANSWER FOR QUESTION 22
```

[57]:
```
import re

def extract_maximum_numeric_value(imput_string):
    pattern = r'\d+'
    matches = re.findall(pattern. input_string)
    if matches:
        max_value = max(map(int, matches))
        return max_value
    else:
        return None
```

[60]:
```
sample_string = 'The maximum value is 1000, but there are also 500 and 750.'
max_value = extract_maximum_numeric_value(sample_string)
print(f"Input string: {sample_string}")
if max_value:
      print(f"Maximum numeric value: {max_value}")
else:
```

```
    print("No numeric value found.")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2224\563402682.py in <module>
      1 sample_string = 'The maximum value is 1000, but there are also 500 and␣
 ↪750.'
----> 2 max_value = extract_maximum_numeric_value(sample_string)
      3 print(f"Input string: {sample_string}")
      4 if max_value:
      5       print(f"Maximum numeric value: {max_value}")

~\AppData\Local\Temp\ipykernel_2224\3198598603.py in␣
 ↪extract_maximum_numeric_value(imput_string)
      3 def extract_maximum_numeric_value(imput_string):
      4     pattern = r'\d+'
----> 5     matches = re.findall(pattern. input_string)
      6     if matches:
      7         max_value = max(map(int, matches))

AttributeError: 'str' object has no attribute 'input_string'
```

[61]:
```
#ANSWER FOR QUESTION 23
```

[70]:
```
import re

def insert_spaces_between_capital_words(input_string):
    pattern = r'(?<!^)(?=[A-Z])'
    spaced_string = re.sub(pattern, ' ', input_string)
    return spaced_string
```

[72]:
```
sample_string = 'TheQuickBrownFoxJumpsOverTheLazyDog'
spaced_string = insert_spaces_between_capital_words(sample_string)
print(f"Input string: {sample_string}")
print(f"Spaced string: {spaced_string}")
```

```
Input string: TheQuickBrownFoxJumpsOverTheLazyDog
Spaced string: The Quick Brown Fox Jumps Over The Lazy Dog
```

[73]:
```
#ANSWER FOR QUESTION 24
```

[74]:
```
import re

def find_sequences(input_string):
    pattern = r'[A-Z][a-z]+'
    matches = re.findall(pattern, input_string)
    return matches
```

```
sample_string = 'TheQuickBrownFoxJumpsOverTheLazyDog'
sequences = find_sequences(sample_string)
print(f"Input string: {sample_string}")
print(f"Sequences: {sequences}")
```

```
Input string: TheQuickBrownFoxJumpsOverTheLazyDog
Sequences: ['The', 'Quick', 'Brown', 'Fox', 'Jumps', 'Over', 'The', 'Lazy',
'Dog']
```

[75]:
```
#ANSWER FOR QUESTION 25
```

[76]:
```
import re

def remove_duplicate_words(sentence):
    pattern = r'\b(\w+)(\s+\1\b)+'
    cleaned_sentence = re.sub(pattern, r'\1', sentence, flags=re.IGNORECASE)
    return cleaned_sentence
```

[77]:
```
sample_sentence = 'The quick brown fox jumps over the lazy dog dog.'
cleaned_sentence = remove_duplicate_words(sample_sentence)
print(f"Original sentence: {sample_sentence}")
print(f"Cleaned sentence: {cleaned_sentence}")
```

```
Original sentence: The quick brown fox jumps over the lazy dog dog.
Cleaned sentence: The quick brown fox jumps over the lazy dog.
```

[79]:
```
#ANSWER FOR QUESTION 26
```

[80]:
```
import re

def accept_string_ending_with_alphanumeric(input_string):
    pattern = r'^.*[a-zA-Z0-9]$'
    match = re.match(pattern, input_string)
    return match is not None
```

[81]:
```
sample_string = 'HelloWorld123'
result = accept_string_ending_with_alphanumeric(sample_string)
print(f"Input string: {sample_string}")
print(f"Accepted: {result}")
```

```
Input string: HelloWorld123
Accepted: True
```

[82]:
```
#ANSWER FOR QUESTION 27
```

[83]:
```
import re

def extract_hashtags(text):
```

```
    pattern = r'\#\w+'
    hashtags = re.findall(pattern, text)
    return hashtags
```

[84]:
```
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'
hashtags = extract_hashtags(sample_text)
print(f"Sample text: {sample_text}")
print(f"Hashtags: {hashtags}")
```

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

[85]:
```
#ANSWER FOR QUESTION 28
```

[86]:
```
import re

def remove_special_symbols(text):
    pattern = r'<U\+[A-Za-z0-9]+\>'
    cleaned_text = re.sub(pattern, '', text)
    return cleaned_text
```

[87]:
```
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"
cleaned_text = remove_special_symbols(sample_text)
print(f"Sample text: {sample_text}")
print(f"Cleaned text: {cleaned_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
Cleaned text: @Jags123456 Bharat band on 28??<ed><ed>Those who  are protesting
#demonetization  are all different party leaders
```

[88]:
```
#ANSWER FOR QUESTION 29
```

[89]:
```
import re

def extract_dates_from_text(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        pattern = r'\d{2}-\d{2}-\d{4}'
        dates = re.findall(pattern, text)
        return dates
```