Regular Expressions ---Regex

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 [1]:
```
import re
def conatins_only_allowed_chars(string):
    pattern = r'^[a-zA-Z0-9]+$'
    match = re.fullmatch(pattern, string)
    return match is not None

string1='Ram has 12 bike.'
string2='Ram_has_12_bike_.'
string3= 'Ramhas12bike'
string4= ' '
print(conatins_only_allowed_chars(string1))
print(conatins_only_allowed_chars(string2))
print(conatins_only_allowed_chars(string3))
print(conatins_only_allowed_chars(string4))
```

```
False
False
True
False
```

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

In [7]:
```
import regex as re
def match_chars():
    string = input('Enter a string: ')
    pattern = '^ab*$'
    matched = re.search(pattern, string)
    if matched != None:
        return 'Matched found : ' + matched.group()
    else:
        return 'No match'
match_chars()
```

```
Enter a string: abbb
```

Out[7]:
```
'Matched found : abbb'
```

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

In [1]:
```
import re
def text_match():
    pattern = '^a(b+)$'
    for i in range(5):
        string=input('Enter a string: ')
        matched = re.search(pattern, string)
        if matched:
            print('Matched found : ' + matched.group() + '\n')
        else:
            print('No match\n')
text_match()
```

```
Enter a string: abb
Matched found :abb

Enter a string: abb123
No match

Enter a string: 123abb
No match

Enter a string: bbaaa
No match

Enter a string: abbbb
Matched found :abbbb
```

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

In [4]:
```python
import re
def text_match(string):
    pattern = '^ab?$'
    matched = re.search(pattern, string)
    if matched:
        return 'Matched found'
    else:
        return 'No match'
print('text_match_1:',text_match('ab'))
print('text_match_2:',text_match('abbc'))
```

```
text_match_1: Matched found
text_match_2: No match
```

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

In [5]:
```python
import re
def text_match(string):
    pattern = '^ab{3}?$'
    matched = re.search(pattern, string)
    if matched:
        return 'Matched found'
    else:
        return 'No match'
print('text_match_1:',text_match('ab'))
print('text_match_2:',text_match('aabbc'))
print('text_match_2:',text_match('abbb'))
```

```
text_match_1: No match
text_match_2: No match
text_match_2: Matched found
```

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

In [12]:
```python
import re

def split_uppercase(string):
    result = re.split(r'([A-Z][^A-Z]*)', string)
    result = [ word for word in result if word]
    return result

split_uppercase('ImportanceOfRegularExpressionsInPython')
```

Out[12]:
```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

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

In [13]:
```
def match_string(x):
    result = re.search(r'a(b{2,3})', x)
    if result:
        return 'Match found: ' + result.group()
    else:
        return 'Match not found '
x1='bbba'
x2='abbbbbb'
print(match_string(x1))
print(match_string(x2))
```

```
Match not found
Match found: abbb
```

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

In [14]:
```
import re
def text_match(text):
        patterns = '^[a-z]+_[a-z]+$'
        result = re.search(patterns,  text)
        if result:
                return 'Found a match: ' + result.group()
        else:
                return('Not matched!')

print(text_match("thisisr_waschairr"))
print(text_match("aab_Abbbc"))
print(text_match("Aaab_abbbc"))
```

```
Found a match: thisisr_waschairr
Not matched!
Not matched!
```

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

In [19]:
```
import re
def text_match(text):
        patterns = 'a.*?b$'
        result = re.search(patterns,  text)
        if result:
                return 'Found a match: ' + result.group()
        else:
                return('Not matched!')

print(text_match("This is a bab"))
print(text_match("This is a bat."))
print(text_match("aaab_abbbb"))
```

```
Found a match: a bab
Not matched!
Found a match: aaab_abbbb
```

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

In [24]:
```
def word_match(string):
    pattern ='^\w+'
    result = re.search(pattern, string)
    if result != None:
        return result.group()
    else:
        return 'No match'
print(word_match('Virat Kohali scored 90 runs in his 100th match'))
print(word_match(' Virat Kohali scored 90 runs in his 100th match'))
```

```
Virat
No match
```

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

In [29]:
```
def match_(string):
    pattern = '\w$'
    result = re.search(pattern, string)
    if result != None:
        return result
    else:
        return 'No match'
print(word_match('Virat Kohali scored 90 runs in his 100th match'))
print(word_match(' virat Kohali scored 90 runs in his 100th match'))
print(word_match('1988 Virat Kohali scored 90 runs in his 100th match'))
print(word_match('virat Kohali scored 90 runs in his 100th match'))
print(word_match('_ virat Kohali scored 90 runs in his 100th match'))
```

```
Virat
No match
1988
virat
_
```

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

In [32]:
```
import re
def match_num(x):
    pattern = re.compile(r'^0')
    result = pattern.search(x)

    if result != None:
        return 'Matched: ' + result.string
    else:
        return  'Non Matched'
print(match_num('0001123'))
print(match_num('123000123'))
print(match_num('000AAA'))
```

```
Matched: 0001123
Non Matched
Matched: 000AAA
```

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

In [33]:
```
def remove_leading_zeros(ip_address):
    pattern = r'\.[0]*'
    modified_ip_address = re.sub(r'\.[0]*','.',ip_address)
    return modified_ip_address
```

```
ip1 = "215.07.049.140"
print(remove_leading_zeros(ip1))
```

```
215.7.49.140
```

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.

In [45]:
```
import re
def extract_dates_from_file(filename):
    with open(filename, 'r') as file:
        text = file.read()

    pattern = r"\b[A-Za-z]+\s+\d{1,2}(?:st|nd|rd|th)?\s+\d{4}\b"
    dates = re.findall(pattern, text)
    return dates
extract_dates_from_file('sample_text.txt')
```

Out[45]:
```
['August 15th 1947']
```

15. Write a Python program to search some literals strings in a string. Go to the editor

In [59]:
```
def match_literals_words(text):
    patterns = ['fox', 'dog', 'horse']
    text = 'The quick brown fox jumps over the lazy dog.'
    for pattern in patterns:
        print('Sreaching for "%s" in "%s" ->' % (pattern, text))
        if re.search(pattern, text):
            print('Matched')
        else:
            print('Not Matched')
match_literals_words(text)
```

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

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

In [60]:
```
def match_literals_char(string):
    pattern = 'fox'
    result = re.search(pattern, string)

    if result !=None:
        return 'Matched found: ' + result.group() + ' in' + ' ' + string + 'from '

    else:
        'Matched not found'
match_literals_char('The quick brown fox jumps over the lazy dog.')
```

Out[60]:
```
'Matched found: fox in The quick brown fox jumps over the lazy dog.from 16 to 19'
```

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

In [61]:
```
def find_substrings(string, pattern):
    substring = []
    p = re.compile(pattern)
```

```
    result = p.findall(string)
    for match in result:
        substring.append(match)

    return substring
input_string = input('Enter a string: ')
input_pattern = input('Enter a pattern: ')

print(find_substrings(input_string, input_pattern))
```

```
Enter a string: Python exercises, PHP exercises, C# exercises
Enter a pattern: exercises
['exercises', 'exercises', 'exercises']
```

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

In [62]:
```
def match_literals_char(string):
    pattern = 'exercises'
    result = re.findall(pattern, string)

    if result !=None:
        return 'Matched found: ' + str(result) + ' in' + ' ' + string

    else:
        'Matched not found'
match_literals_char('Python exercises, PHP exercises, C# exercises')
```

Out[62]:
```
"Matched found: ['exercises', 'exercises', 'exercises'] in Python exercises, PHP e
xercises, C# exercises"
```

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

In [64]:
```
def convert_date_format(date):
    pattern = r'(\d{4})-(\d{2})-(\d{2})'
    converted_date = re.sub(pattern, '\\3-\\2-\\1', date)
    return converted_date

print('New date format dd-mm-yyyy : {}'.format(convert_date_format("2023-07-20")))
```

```
New date format dd-mm-yyyy : 20-07-2023
```

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

In [65]:
```
import re

def find_word_strating_with_a_or_e(text):
    pattern = (r"\b(?:a|e)\w+\b")
    result = re.findall(pattern, text, re.IGNORECASE)

    return result
print('Find a word start with a and e : ', find_word_strating_with_a_or_e("The fol
```

```
Find a word start with a and e :  ['example', 'an', 'ArrayList', 'elements', 'elem
ents', 'are', 'added', 'ArrayList', 'and', 'ArrayList', 'accordingly']
```

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

In [66]:
```
import re

def separate_number_and_position(text):
    p = re.compile(r'\d+')
    result = p.finditer(text)
    for m in result:
        print('Number : ', str(m.group(0)),", Start_Index:", str(m.start()) + ', E
separate_number_and_position('Ten 10, Twenty 20, Thirty 30.')
```

```
Number :  10 , Start_Index: 4, End_Index : 6
Number :  20 , Start_Index: 15, End_Index : 17
Number :  30 , Start_Index: 26, End_Index : 28
```

22. Write a regular expression in python program to extract maximum numeric value from a string

In [67]:
```
import re
def extract_max_num_value(text):
    p = re.compile(r'\d+')
    nums = p.findall(text)
    max_num = max(map(int, nums))
    return max_num

extract_max_num_value('The maximum numerical value is 42 and the minimum is 1')
```

Out[67]:
```
42
```

23. Write a Regex in Python to put spaces between words starting with capital letters.

In [68]:
```
def put_space_for_capital_letters(text):
    pattern = r'(?=[A-Z])'
    result = re.sub(pattern, " ", text)
    return result
put_space_for_capital_letters('ThisIsAPen.')
```

Out[68]:
```
' This Is A Pen.'
```

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

In [70]:
```
import re
def upper_follow_lower_char(text):
    pattern = r'[A-Z]+[a-z]+'
    matches = re.findall(pattern, text)
    return bool(matches)
print(upper_follow_lower_char('AaBCdef'))
print(upper_follow_lower_char('ThisIsAPen'))
print(upper_follow_lower_char('Hello World, OpenAI is Awesome'))
```

```
True
True
True
```

25.Write a Python program to remove duplicate words from Sentence using Regular Expression

In [76]:
```
import regex as re
def remove_duplicate_word(text):
    pattern= r'\b(\w+)(?:\W+\1\b)+'
```

```
    x = re.sub(pattern, r'\1', text)
    return x
remove_duplicate_word('Ram went went to his his home.')
```

Out[76]:
```
'Ram went to his home.'
```

26. Write a python program using RegEx to accept string ending with alphanumeric character.

In [77]:
```
import re
def ends_with_alphanumeric():
    pattern = r'[a-zA-Z0-9]$'

    for i in range(5):
        string=input('Enter a string: ')
        match =re.search(pattern, string)
        if match:
            print('String ends with an alphanumeric character.\n')
        else:
            print('String does not end with an alphanumeric character.\n')
ends_with_alphanumeric()
```

```
Enter a string: 123bbndc
String ends with an alphanumeric character.

Enter a string: Virat is a cricket player who was born on November 5, 1988.
String does not end with an alphanumeric character.

Enter a string: Ram went to his home number123.
String does not end with an alphanumeric character.

Enter a string: ankitrai326
String ends with an alphanumeric character.

Enter a string: ankirai@
String does not end with an alphanumeric character.
```

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

In [78]:
```
def hash_tags(text):
    pattern = r"#\w+"
    hashtags = re.findall(pattern, text)
    return hashtags

hash_tags("RT @kapil_kausik: #Doltiwal I mean #xyzabc is hurt by #Demonetization as
```

Out[78]:
```
['#Doltiwal', '#xyzabc', '#Demonetization']
```

28. Write a python program using RegEx to remove <U+..> like symbols

In [79]:
```
import re
def remove_unicode_symbol(text):
    pattern = r'<U\+[A-Fa-f0-9]{4}>'
    new_text = re.sub(pattern, '', text)
    return new_text

remove_unicode_symbol("@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00
```

Out[79]:
```
'@Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization a
re all different party leaders'
```

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 [81]:
```
import re
def extract_dates_from_file(filename):
    with open(filename, 'r') as file:
        text = file.read()

    pattern = r"\d{2}-\d{2}-\d{4}"
    dates = re.findall(pattern, text)
    return dates
extract_dates_from_file('sample_text.txt')
```

Out[81]:
```
['12-09-1992', '15-12-1999']
```

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 [4]:
```
def repalce_characters(string):
    pattern = r'[ ,.]'
    replacement = ':'
    modified_string =re.sub(pattern, replacement, string)
    return modified_string

repalce_characters('Python Exercises, PHP exercises.')
```

Out[4]:
```
'Python:Exercises::PHP:exercises:'
```