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 check_string_characters(string):
    pattern = r'^[a-zA-Z0-9]+$'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["abc123", "AbCdEfG", "123456", "abc!@#"]
for string in strings:
    if check_string_characters(string):
        print(f"The string '{string}' consists only of characters from a-z, A-Z, ar
    else:
        print(f"The string '{string}' contains characters other than a-z, A-Z, and
```

```
The string 'abc123' consists only of characters from a-z, A-Z, and 0-9.
The string 'AbCdEfG' consists only of characters from a-z, A-Z, and 0-9.
The string '123456' consists only of characters from a-z, A-Z, and 0-9.
The string 'abc!@#' contains characters other than a-z, A-Z, and 0-9.
```

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

In [3]:

```
import re

def match_string(string):
    pattern = r'ab*'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["ab", "abb", "a", "ac", "abc"]
for string in strings:
    if match_string(string):
        print(f"The string '{string}' satisfies the condition of having an 'a' fol
    else:
        print(f"The string '{string}' does not meet the condition of having an 'a'
```

```
The string 'ab' satisfies the condition of having an 'a' followed by zero or more
'b's.
The string 'abb' satisfies the condition of having an 'a' followed by zero or more
'b's.
The string 'a' satisfies the condition of having an 'a' followed by zero or more
'b's.
The string 'ac' satisfies the condition of having an 'a' followed by zero or more
'b's.
The string 'abc' satisfies the condition of having an 'a' followed by zero or more
'b's.
```

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

In [5]:
```
import re

def match_string(string):
    pattern = r'ab+'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["ab", "abb", "a", "ac", "abc"]
for string in strings:
    if match_string(string):
        print("The string '{}' satisfies the condition of having an 'a' followed by
    else:
        print("The string '{}' does not meet the condition of having an 'a' followe
```

```
The string 'ab' satisfies the condition of having an 'a' followed by one or more
'b's.
The string 'abb' satisfies the condition of having an 'a' followed by one or more
'b's.
The string 'a' does not meet the condition of having an 'a' followed by one or mor
e 'b's.
The string 'ac' does not meet the condition of having an 'a' followed by one or mo
re 'b's.
The string 'abc' satisfies the condition of having an 'a' followed by one or more
'b's.
```

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

In [7]:
```
import re

def match_string(string):
    pattern = r'ab?'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["ab", "a", "abb", "ac", "abc"]
for string in strings:
    if match_string(string):
        print("The string '{}' satisfies the condition of having an 'a' followed by
    else:
        print("The string '{}' does not meet the condition of having an 'a' followe
```

```
The string 'ab' satisfies the condition of having an 'a' followed by zero or one
'b'.
The string 'a' satisfies the condition of having an 'a' followed by zero or one
'b'.
The string 'abb' satisfies the condition of having an 'a' followed by zero or one
'b'.
The string 'ac' satisfies the condition of having an 'a' followed by zero or one
'b'.
The string 'abc' satisfies the condition of having an 'a' followed by zero or one
'b'.
```

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

In [9]:
```
import re

def match_string(string):
    pattern = r'a{1}b{3}'
    match = re.search(pattern, string)
    return match is not None
```

```
# Test the function
strings = ["abbb", "a", "abb", "abbbb", "abc"]
for string in strings:
    if match_string(string):
        print("The string '{}' satisfies the condition of having an 'a' followed by
    else:
        print("The string '{}' does not meet the condition of having an 'a' followe
```

```
The string 'abbb' satisfies the condition of having an 'a' followed by three 'b's.
The string 'a' does not meet the condition of having an 'a' followed by three
'b's.
The string 'abb' does not meet the condition of having an 'a' followed by three
'b's.
The string 'abbbb' satisfies the condition of having an 'a' followed by three
'b's.
The string 'abc' does not meet the condition of having an 'a' followed by three
'b's.
```

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 [11]:
```
import re

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

```
['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 [12]:
```
import re

def match_string(string):
    pattern = r'ab{2,3}'
    match = re.search(pattern, string)
    return match is not None

# Test the function
strings = ["ab", "abb", "abbb", "a", "abc", "abbbb"]
for string in strings:
    if match_string(string):
        print("The string '{}' satisfies the condition of having an 'a' followed by
    else:
        print("The string '{}' does not meet the condition of having an 'a' followe
```

```
The string 'ab' does not meet the condition of having an 'a' followed by two to th
ree 'b's.
The string 'abb' satisfies the condition of having an 'a' followed by two to three
'b's.
The string 'abbb' satisfies the condition of having an 'a' followed by two to thre
e 'b's.
The string 'a' does not meet the condition of having an 'a' followed by two to thr
ee 'b's.
The string 'abc' does not meet the condition of having an 'a' followed by two to t
hree 'b's.
The string 'abbbb' satisfies the condition of having an 'a' followed by two to thr
ee 'b's.
```

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

In [13]:
```
import re

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

# Test the function
text = "hello_world, python_program, open_ai, chat_gpt"
result = find_sequences(text)
print("The sequences of lowercase letters joined with an underscore in the given te
for sequence in result:
    print(sequence)
```

```
The sequences of lowercase letters joined with an underscore in the given text ar
e:
hello_world
python_program
open_ai
chat_gpt
```

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

In [14]:
```
import re

def match_string(string):
    pattern = r'a.*b$'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["ab", "acb", "abc", "a1234b", "adeb", "a_123_b"]
for string in strings:
    if match_string(string):
        print("The string '{}' satisfies the condition of starting with 'a', follow
    else:
        print("The string '{}' does not satisfy the condition of starting with 'a',
```

```
The string 'ab' satisfies the condition of starting with 'a', followed by anythin
g, and ending with 'b'.
The string 'acb' satisfies the condition of starting with 'a', followed by anythin
g, and ending with 'b'.
The string 'abc' does not satisfy the condition of starting with 'a', followed by
anything, and ending with 'b'.
The string 'a1234b' satisfies the condition of starting with 'a', followed by anyt
hing, and ending with 'b'.
The string 'adeb' satisfies the condition of starting with 'a', followed by anythi
ng, and ending with 'b'.
The string 'a_123_b' satisfies the condition of starting with 'a', followed by any
thing, and ending with 'b'.
```

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

In [16]:
```
import re

def match_word(string, word):
    pattern = r'^' + re.escape(word)
    match = re.search(pattern, string)
    return match is not None

# Test the function
string = "java is a powerful programming language."
```

```
words = ["java", "is", "a", "programming", "language"]
for word in words:
    if match_word(string, word):
        print("The word '{}' is found at the beginning of the string.".format(word
    else:
        print("The word '{}' is not found at the beginning of the string.".format(
```

```
The word 'java' is found at the beginning of the string.
The word 'is' is not found at the beginning of the string.
The word 'a' is not found at the beginning of the string.
The word 'programming' is not found at the beginning of the string.
The word 'language' is not found at the beginning of the string.
```

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

In [19]:

```
import re

def match_string(string):
    pattern = r'^[a-zA-Z0-9_]+$'
    match = re.match(pattern, string)
    return match is not None

# Test the function
strings = ["Hello_World123", "Test123", "abc_def", "Abc123_", "Special$Chars"]
for string in strings:
    if match_string(string):
        print("The string '{}' contains only upper and lowercase letters, numbers,
    else:
        print("The string '{}' does not contain only upper and lowercase letters,
```

```
The string 'Hello_World123' contains only upper and lowercase letters, numbers, an
d underscores.
The string 'Test123' contains only upper and lowercase letters, numbers, and under
scores.
The string 'abc_def' contains only upper and lowercase letters, numbers, and under
scores.
The string 'Abc123_' contains only upper and lowercase letters, numbers, and under
scores.
The string 'Special$Chars' does not contain only upper and lowercase letters, numb
ers, and underscores.
```

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

In [20]:

```
def starts_with_number(string, number):
    return string.startswith(str(number))

# Test the function
strings = ["123abc", "456xyz", "789pqr", "987abc", "654xyz"]
number = 123
for string in strings:
    if starts_with_number(string, number):
        print("The string '{}' starts with the number {}.".format(string, number))
    else:
        print("The string '{}' does not start with the number {}.".format(string,
```

```
The string '123abc' starts with the number 123.
The string '456xyz' does not start with the number 123.
The string '789pqr' does not start with the number 123.
The string '987abc' does not start with the number 123.
The string '654xyz' does not start with the number 123.
```

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

In [21]:
```
def remove_leading_zeros(ip_address):
    ip_parts = ip_address.split('.')
    ip_parts = [str(int(part)) for part in ip_parts]
    return '.'.join(ip_parts)

# Test the function
ip_address = "191.168.021.002"
result = remove_leading_zeros(ip_address)
print("Modified IP address: {}".format(result))
```

```
Modified IP address: 191.168.21.2
```

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 Hint- Use re.match() method here

In [26]:
```
import re

def find_date_string(text):
    pattern = r"(January|February|March|April|May|June|July|August|September|Octobe
    match = re.match(pattern, text)
    if match:
        return match.group()
    else:
        return None

# Test the function
text = "On August 15th 1947 that India was declared independent from British coloni
date_string = find_date_string(text)

if date_string:
    print("Matching date string found: {}".format(date_string))
else:
    print("No matching date string found.")
```

```
No matching date string found.
```

15:Write a Python program to search some literals strings in a string. Go to the editor Sample text : 'The quick brown fox jumps over the lazy dog.' Searched words : 'fox', 'dog', 'horse'

In [29]:
```
def search_words(text, words):
    found_words = []
    for word in words:
        if word in text:
            found_words.append(word)
    return found_words

# Test the function
text = 'The quick brown fox jumps over the lazy dog .'
searched_words = ['fox', 'dog', 'horse']
found_words = search_words(text, searched_words)

if found_words:
    print("Found words:")
    for word in found_words:
        print(word)
else:
    print("No words found.")
```

```
Found words:
fox
dog
```

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 [30]:
```
def search_word_with_location(text, word):
    found_locations = []
    index = 0
    while index < len(text):
        found_index = text.find(word, index)
        if found_index != -1:
            found_locations.append(found_index)
            index = found_index + 1
        else:
            break
    return found_locations

# Test the function
text = 'The quick brown fox jumps over the lazy dog.'
searched_word = 'fox'
found_locations = search_word_with_location(text, searched_word)

if found_locations:
    print("Found word '{}' at locations:".format(searched_word))
    for location in found_locations:
        print(location)
else:
    print("Word '{}' not found.".format(searched_word))
```

```
Found word 'fox' at locations:
16
```

17:Write a Python program to find the substrings within a string. Sample text : 'Python exercises, PHP exercises, C# exercises' Pattern : 'exercises'.

In [32]:
```
import re

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

# Test the function
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
substrings = find_substrings(text, pattern)

if substrings:
    print("Found substrings:")
    for substring in substrings:
        print(substring)
else:
    print("No substrings found.")
```

```
Found substrings:
exercises
exercises
exercises
```

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

In [33]:
```
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

# Test the function
text = 'Python exercises, PHP exercises, C# exercises'
substring = 'exercises'
occurrences = find_substring_occurrences(text, substring)

if occurrences:
    print("Substring occurrences:")
    for substring, index in occurrences:
        print(f"'{substring}' found at position {index}")
else:
    print("No occurrences found.")
```

```
Substring occurrences:
'exercises' found at position 7
'exercises' found at position 22
'exercises' found at position 36
```

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

In [35]:
```
def convert_date_format(date):
    parts = date.split('-')
    if len(parts) != 3:
        return None
    return f"{parts[2]}-{parts[1]}-{parts[0]}"

# Test the function
date = "2019-12-31"
converted_date = convert_date_format(date)

if converted_date:
    print("Original date: {}".format(date))
    print("Converted date: {}".format(converted_date))
else:
    print("Invalid date format.")
```

```
Original date: 2019-12-31
Converted date: 31-12-2019
```

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

In [37]:
```
def find_words_starting_with_a_or_e(text):
    words = []
    for word in text.split():
        if word.startswith('a') or word.startswith('e'):
            words.append(word)
    return words
```

```
# Test the function
text = "Apple is an amazing fruit. Elephants are enormous creatures. Ants are tiny
words = find_words_starting_with_a_or_e(text)

if words:
    print("Words starting with 'a' or 'e':")
    for word in words:
        print(word)
else:
    print("No words found.")
```

```
Words starting with 'a' or 'e':
an
amazing
are
enormous
are
```

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

In [38]:

```
import re

def separate_numbers_with_position(text):
    pattern = r"\d+"
    matches = re.finditer(pattern, text)

    for match in matches:
        number = match.group()
        position = match.start()
        print(f"Number: {number}, Position: {position}")

# Test the function
text = "Hello 123 world 456!"
separate_numbers_with_position(text)
```

```
Number: 123, Position: 6
Number: 456, Position: 16
```

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

In [39]:

```
import re

def extract_max_numeric_value(text):
    pattern = r"\d+"
    numbers = re.findall(pattern, text)

    if numbers:
        max_number = max(numbers, key=int)
        return max_number
    else:
        return None

# Test the function
text = "The maximum value is 9876 among the numbers 123, 456, and 9876."
max_value = extract_max_numeric_value(text)

if max_value:
    print("Maximum numeric value: {}".format(max_value))
else:
    print("No numeric value found.")
```

```
Maximum numeric value: 9876
```

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

In [41]:
```python
import re

def insert_spaces(text):
    pattern = r"(?<!\b[A-Z])\B([A-Z])"
    modified_text = re.sub(pattern, r' \1', text)
    return modified_text

# Test the function
text = "ThisIsARegexExample"
modified_text = insert_spaces(text)
print("Modified text:", modified_text)
```

```
Modified text: This Is A Regex Example
```

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

In [42]:
```python
import re

def find_sequences(text):
    pattern = r"[A-Z][a-z]+"
    sequences = re.findall(pattern, text)
    return sequences

# Test the function
text = "This is a Sample Text with Multiple Sequences like ThisOne and AnotherOne"
sequences = find_sequences(text)

if sequences:
    print("Sequences found:")
    for sequence in sequences:
        print(sequence)
else:
    print("No sequences found.")
```

```
Sequences found:
This
Sample
Text
Multiple
Sequences
This
One
Another
One
```

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

In [44]:
```python
import re

def remove_duplicate_words(sentence):
    pattern = r'\b(\w+)\b\s+(?=.*\b\1\b)'
    modified_sentence = re.sub(pattern, '', sentence)
    return modified_sentence.strip()

# Test the function
sentence = "This is a sentence with four words, sentence and four words."
```

```
modified_sentence = remove_duplicate_words(sentence)
print("Modified sentence:", modified_sentence)
```

```
Modified sentence: This is a with words, sentence and four words.
```

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

In [46]:
```
import re

def accept_string_ending_with_alphanumeric(text):
    pattern = r"\w$"
    match = re.search(pattern, text)
    if match:
        return True
    else:
        return False

# Test the function
text1 = "print146"
text2 = "ABC-146_"
text3 = "Python@"

print(accept_string_ending_with_alphanumeric(text1))  # True
print(accept_string_ending_with_alphanumeric(text2))  # True
print(accept_string_ending_with_alphanumeric(text3))  # False
```

```
True
True
False
```

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 <U+00A0><U+00BD><U+00B1><U+0089> "acquired funds" No wo""" Output: ['#Doltiwal', '#xyzabc', '#Demonetization']

In [47]:
```
import re

def extract_hashtags(text):
    pattern = r"#\w+"
    hashtags = re.findall(pattern, text)
    return hashtags

# Test the function
text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization a
hashtags = extract_hashtags(text)

if hashtags:
    print("Extracted hashtags:")
    for hashtag in hashtags:
        print(hashtag)
else:
    print("No hashtags found.")
```

```
Extracted hashtags:
#Doltiwal
#xyzabc
#Demonetization
```

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 all such symbols. Sample

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

In [48]:
```
import re

def remove_special_symbols(text):
    pattern = r"<U\+[\w\s]+>"
    modified_text = re.sub(pattern, '', text)
    return modified_text

# Test the function
text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Thos
modified_text = remove_special_symbols(text)
print("Modified text:", modified_text)
```

```
Modified text: @Jags123456 Bharat band on 28??<ed><ed>Those who  are protesting #d
emonetization  are 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 [50]:
```
import sys
import re

def extract_dates_from_file(filename):
    pattern = r"\b\d{2}-\d{2}-\d{4}\b"
    dates = []

    with open(filename, 'r') as file:
        text = file.read()
        dates = re.findall(pattern, text)

    return dates

# Test the function
if len(sys.argv) < 2:
    print("Please provide the filename as a command-line argument.")
else:
    filename = sys.argv[1]
    dates = extract_dates_from_file(filename)

    if dates:
        print("Dates found:")
        for date in dates:
            print(date)
    else:
        print("No dates found.")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11544\3331148669.py in <module>
     17 else:
     18     filename = sys.argv[1]
---> 19     dates = extract_dates_from_file(filename)
     20 
     21     if dates:

~\AppData\Local\Temp\ipykernel_11544\3331148669.py in extract_dates_from_file(file
name)
      6     dates = []
      7 
----> 8     with open(filename, 'r') as file:
      9         text = file.read()
     10         dates = re.findall(pattern, text)

FileNotFoundError: [Errno 2] No such file or directory: '-f'
```

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 [51]:
```python
def replace_space_comma_dot_with_colon(text):
    modified_text = text.replace(' ', ':').replace(',', ':').replace('.', ':')
    return modified_text

# Test the function
text = 'Python Exercises, PHP exercises.'
modified_text = replace_space_comma_dot_with_colon(text)
print("Modified text:", modified_text)
```

```
Modified text: Python:Exercises::PHP:exercises:
```

In [ ]: