Python Regular Expressions RegEx

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

RegEx Module Python has a built-in package called re, which can be used to work with Regular Expressions.

Import Regular Expression Library

In [1]:

```
import re
```

create a pattern object representing a compiled regular expression using re.compile() method. Add the followign text to it:
'Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?'

In [10]:

```
# Define the text
text = 'Hello, I am studying Deep Learning at Mt. SAC. How are you doing?'

# Create a pattern object using re.compile()
pattern = re.compile('Deep Learning')
```

print the pattern

In [13]:

```
# Print the pattern
print("Pattern:", pattern)
```

```
Pattern: re.compile('Deep Learning')
```

Using the search method, find the words 'Deep Learning' in the followign text:
'Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?'

In [18]:

```
# Use the search method to find 'Deep Learning' in the text
match = pattern.search(text)

# Check if the pattern was found
if match:
    print(f"Found '{match.group()}' in the text!")
else:
    print("Pattern not found.")
```

```
Found 'Deep Learning' in the text!
```

Given the followign text, find the word 'NLP' and print the output. Hint: use re.searc() and match.group()

In [20]:

```
match = re.search('NLP', 'In this class, we will do several labs to understand NLP'
# enter your code here:

# Check if the pattern was found
if match:
    # Print the found word
    print(f"Found: {match.group()}")
else:
    print("Pattern not found.")
```

```
Found: NLP
```

Write a function to match regular expression patterns: def find_pattern:

In [26]:

```
def find_pattern(pattern, text):
    # Compile the pattern into a regex object
    regex = re.compile(pattern)

    # Search for the pattern in the text
    match = regex.search(text)

    # Check if the pattern was found
    if match:
        # Return the matched string
        return match.group()
    else:
        # Return None if no match was found
        return None

# Example usage:
pattern = 'NLP'
text = 'In this class, we will do several labs to understand NLP'
result = find_pattern(pattern, text)

if result:
    print(f"Found: {result}")
else:
    print("Pattern not found.")
```

```
Found: NLP
```

Call the function find_pattern with the following values:

- pattern: We are learning NLP, so we can program a chatbot before the end of teh semester.
- text: chatbot

In [33]:

```
pattern = "chatbot"
text = "We are learning NLP, so we can program a chatbot before the end of the seme

# Call the function
```

```
result = find_pattern(pattern, text)

# Print the result
if result:
    print(f"Found: {result}")
else:
    print("Pattern not found.")
```

```
Found: chatbot
```

What is regex?

Regular expressions, often abbreviated as "regex," are sequences of characters that form search patterns. They are used to match, search, and manipulate strings in a flexible and powerful way. Regex is commonly used for validating input, searching for patterns within text, and parsing strings.

Using the split() function, separate the words of the following text

In [38]:

```
txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"
```

In [42]:

```
txt = "Hello, I am studying Deep Learning at Mt. SAC. How are you doing?"

# Split the text into words
words = txt.split()

# Print the list of words
print(words)
```

```
['Hello,', 'I', 'am', 'studying', 'Deep', 'Learning', 'at', 'Mt.', 'SAC.', 'How', 'a
re', 'you', 'doing?']
```

Using the sub() Function, replace all the white-spaces with the number "9"

In [ ]:

```
txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"
```

In [45]:

```
# Use re.sub() to replace all white spaces with "9"
result = re.sub(r'\s', '9', txt)

# Print the result
print(result)
```

```
Hello,9I9am9studying9Deep9Learning9at9Mt.9SAC.9How9are9you9doing?
```

You can control the number of replacements by specifying the count parameter.

replace the first two white spaces in the followign text with the number 9 and print the new text

In [ ]:

```
txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"
```

In [47]:

```
# Replace the first two white spaces with "9"
result = re.sub(r'\s', '9', txt, count=2)

# Print the new text
print(result)
```

```
Hello,9I9am studying Deep Learning at Mt. SAC. How are you doing?
```

What are the five most important re functions that you would be required to use most of the times are

The five most important functions in Python's re module that would likely use most of the time are:

re.search(): Scans through a string and looks for the first location where the regular expression pattern produces a match. re.match(): Determines if the regular expression matches at the beginning of the string. re.findall(): Finds all occurrences of a pattern in a string and returns them as a list. re.sub(): Substitutes occurrences of the pattern in a string with a replacement string. re.compile(): Compiles a regular expression pattern into a regex object, which can be used for pattern matching.

The Match object has properties and methods used to retrieve information about the search, and the result:

- span() returns a tuple containing the start-, and end positions of the match.
- string returns the string passed into the function
- group() returns the part of the string where there was a match

Using the span method, find the start and end position of the word Deep. Hint, look for the first word that has the letter "D"

In [51]:

```
txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"
# Enter your code here
# Search for the first word that has the letter "D"
match = re.search(r'\bD\w+', txt)

# Check if a match was found
if match:
    # Use span() method to find the start and end positions of the match
    start, end = match.span()
    print(f"The word '{match.group()}' starts at position {start} and ends at posit
else:
    print("No match found.")
```

```
The word 'Deep' starts at position 20 and ends at position 24.
```

Display the first word that starts with the letter "D"

In [55]:

```python
txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"
# Enter your code here
# Search for the first word that starts with the letter "D"
match = re.search(r'\bD\w+', txt)

# Check if a match was found
if match:
    # Display the first word that starts with "D"
    print(f"The first word that starts with 'D' is: {match.group()}")
else:
    print("No word starting with 'D' found.")
```

```
The first word that starts with 'D' is: Deep
```

Create pattern object with the word "Deep"

In [74]:

```python
# Create a pattern object with the word "Deep"
pattern = re.compile("Deep")

# Print the pattern object
print(f"Pattern object created: {pattern}")
```

```
Pattern object created: re.compile('Deep')
```

Create a function named: "highlight_regex_matches" to highlight words wiht green color

In [77]:

```python
#from colorama import Back, Style
#def highlight_regex_matches(pattern, text, print_output=True):
    #complete the function

from colorama import Fore, Style
import re

def highlight_regex_matches(pattern, text, print_output=True):
    """
    Highlights matches of the given pattern in the text with green color.

    Args:
    - pattern (str): The regex pattern to search for.
    - text (str): The text in which to search for the pattern.
    - print_output (bool): Whether to print the output with highlights.

    Returns:
    - str: The text with highlighted matches if print_output is False.
    """
    # Compile the pattern
    regex = re.compile(pattern)

    # Use re.finditer to find all matches
    matches = list(regex.finditer(text))

    # Create a highlighted version of the text
    highlighted_text = text
    offset = 0
    for match in matches:
        start, end = match.span()
        highlighted_text = (highlighted_text[:start + offset] +
                            Fore.GREEN + highlighted_text[start + offset:end + offs
                            highlighted_text[end + offset:])
        offset += len(Fore.GREEN) + len(Style.RESET_ALL)

    if print_output:
        print(highlighted_text)
    else:
        return highlighted_text

# Example usage
pattern = "Deep"
text = "Hello, I am studying Deep Learning at Mt. SAC. How are you doing?"
highlight_regex_matches(pattern, text)
```

```
Hello, I am studying Deep Learning at Mt. SAC. How are you doing?
```

call the function with these parameters (pattern, "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?")
Make sure teh word "Deep" is highlighted in green.

In [88]:

```
from IPython.display import HTML

def highlight_regex_matches_html(pattern, text):
    """
    Highlights matches of the given pattern in the text with green background color

    Args:
    - pattern (str): The regex pattern to search for.
    - text (str): The text in which to search for the pattern.

    Returns:
    - str: The HTML formatted text with highlighted matches.
    """
    import re

    # Compile the pattern
    regex = re.compile(pattern)

    # Create a highlighted version of the text
    highlighted_text = text
    offset = 0

    # Find all matches and highlight them
    for match in regex.finditer(text):
        start, end = match.span()
        highlighted_text = (highlighted_text[:start + offset] +
                            '<span style="background-color: green; color: white;">'
                            highlighted_text[end + offset:])
        offset += len('<span style="background-color: green; color: white;">') + le

    return highlighted_text

# Parameters
```

```
pattern = "Deep"
text = "Hello, I am studying Deep Learning at Mt. SAC. How are you doing?"

# Display highlighted text in Jupyter Notebook
HTML(highlight_regex_matches_html(pattern, text))
```

Out[88]: Hello, I am studying Deep Learning at Mt. SAC. How are you doing?

In [15]:

```
Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?
```

Consider an example below where we have `sentence` and `sentense` spellings. We want to find all occurances of both words in the text.

In [16]:

```
txt = """when a judge sentences someone, they officially state what someone’s punis
He was sentenced to 15 years in prison, but the officials reduce the sentense"""
```

define the pattern

In [121…

```
from IPython.display import HTML
import re

def highlight_regex_matches_html(pattern, text):
    """
    Highlights matches of the given pattern in the text with a green background col

    Args:
    - pattern (str): The regex pattern to search for.
    - text (str): The text in which to search for the pattern.

    Returns:
    - str: The HTML formatted text with highlighted matches.
    """
    # Compile the pattern
    regex = re.compile(pattern)

    # Create a highlighted version of the text
    highlighted_text = text
    offset = 0

    # Find all matches and highlight them
    for match in regex.finditer(text):
        start, end = match.span()
        highlighted_text = (highlighted_text[:start + offset] +
                            '<span style="background-color: green; color: white;">'
                            highlighted_text[end + offset:])
        offset += len('<span style="background-color: green; color: white;">') + le
```

find all the words in the text that match the pattern

In [123…

```
import re

# Define the text
```

```
txt = """when a judge sentences someone, they officially state what someone’s punis
He was sentenced to 15 years in prison, but the officials reduce the sentense"""

# Define the pattern to match 'sentence', 'sentences', 'sentenced', and 'sentense'
pattern = r'\bsentences?\b|\bsentenced\b|\bsentense\b'

# Compile the pattern
regex = re.compile(pattern)

# Find all matches
matches = regex.findall(txt)

# Display the matches
print(matches)
```

```
['sentences', 'sentenced', 'sentense']
```

In [18]:

```
```

Out[18]:

```
['sentence', 'sentence', 'sentense']
```

Highlight all the words that match the pattern

In [125…

```
from IPython.display import HTML

def highlight_regex_matches_html(pattern, text):
    """
    Highlights matches of the given pattern in the text with green background color

    Args:
    - pattern (str): The regex pattern to search for.
    - text (str): The text in which to search for the pattern.

    Returns:
    - str: The HTML formatted text with highlighted matches.
    """
    import re

    # Compile the pattern
    regex = re.compile(pattern)

    # Create a highlighted version of the text
    highlighted_text = text
    offset = 0

    # Find all matches and highlight them
    for match in regex.finditer(text):
        start, end = match.span()
        highlighted_text = (highlighted_text[:start + offset] +
                            '<span style="background-color: green; color: white;">'
                            highlighted_text[end + offset:])
        offset += len('<span style="background-color: green; color: white;">') + le

    return highlighted_text

# Define the text
```

```
txt = """when a judge sentences someone, they officially state what someone’s punis
He was sentenced to 15 years in prison, but the officials reduce the sentense"""

# Define the pattern to match 'sentence', 'sentences', 'sentenced', and 'sentense'
pattern = r'\bsentences?\b|\bsentenced\b|\bsentense\b'

# Display highlighted text in Jupyter Notebook
HTML(highlight_regex_matches_html(pattern, txt))
```

Out[125…

when a judge sentences someone, they officially state what someone’s punishment will be.
He was sentenced to 15 years in prison, but the officials reduce the sentense

In [19]:

```
when a judge sentences someone, they officially state what someone’s punishment w
ill be.
He was sentenced to 15 years in prison, but the officials reduce the sentense
```

List 5 Meta sequences

Meta-sequences in regular expressions are special sequences that provide functionality beyond basic character matching. Here are five commonly used meta-sequences:

1. . (Dot): Matches any single character except for newline characters.

Example: a.c matches "abc", "a1c", "a-c", etc., but not "ac" or "a\nc".

2. \d: Matches any digit (equivalent to [0-9]).

Example: \d{2} matches "12", "99", "00", etc.

3. \w: Matches any alphanumeric character (letters and digits) and underscore (equivalent to [a-zA-Z0-9_]).

Example: \w{5} matches "hello", "12345", "abc_1", etc.

4. \s: Matches any whitespace character (spaces, tabs, and newlines).

Example: \s+ matches one or more whitespace characters, such as " ", "\t", or "\n".

5. ^ and $: Anchors that match the start and end of a string, respectively.

^: Matches the start of a string. Example: ^abc matches "abc" at the beginning of the string.
$: Matches the end of a string. Example: xyz$ matches "xyz" at the end of the string.

Read the following text

In [132…

```
txt ="""American football evolved in the United States, originating from the sports
The first American football match was played on November 6, 1869, between two colle
using rules based on the rules of soccer at the time.

A set of rule changes drawn up from 1880 onward by Walter Camp,
the "Father of American Football", established the snap, the line of scrimmage,
eleven-player teams, and the concept of downs! Later rule changes legalized the for
created the neutral zone and specified the size and shape of the football.

The sport is closely related to Canadian football!
It evolved in parallel with and at the same time as the American game,
although its rules were developed independently from those of Camp.

Most of the features that distinguish American football from rugby and soccer are a
The two sports are considered the primary variants of gridiron football!

American football is the most popular sport in the United States!

The most popular forms of the game are professional and college football,
with the other major levels being high school and youth football.
As of 2012, nearly 1.1 million high school athletes and 70,000 college athletes pla
"""
```

Find all the years listed in the text and display them

In [135…

```
# Define the pattern to match years
pattern = r'\b\d{4}\b'

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

# Display the years
print(years)
```

```
['1869', '1880', '2012']
```

In [21]:

Out[21]:

```
['1869', '1880', '2012']
```

Find all the sentences which do not end with a full stop (.) in the given text.

In [137…

```
# Define the pattern to match sentences that do not end with a full stop
pattern = r'[^.!?]*[^.!?]\n'

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

# Remove any empty sentences
sentences = [s.strip() for s in sentences if s.strip()]

# Display the sentences
for sentence in sentences:
    print(sentence)
```

```
The first American football match was played on November 6, 1869, between two colleg
e teams, Rutgers and Princeton,
A set of rule changes drawn up from 1880 onward by Walter Camp,
the "Father of American Football", established the snap, the line of scrimmage,
Later rule changes legalized the forward pass,
It evolved in parallel with and at the same time as the American game,
The most popular forms of the game are professional and college football,
```

In [ ]:

Highlight those sentences

In [141…

```python
from IPython.display import HTML
import re

def highlight_sentences_no_full_stop(text):
    """
    Highlights sentences in the text that do not end with a full stop.

    Args:
    - text (str): The text in which to search for sentences.

    Returns:
    - str: The HTML formatted text with highlighted sentences.
    """
    # Define the pattern to match sentences that do not end with a full stop
    pattern = r'([^\n.!?]*[^.!?\n][^\n]*)'

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

    # Create a highlighted version of the text
    highlighted_text = text
    for sentence in sentences:
        # Only highlight non-empty sentences that do not end with a full stop
        if sentence.strip() and not sentence.strip().endswith(('.', '!', '?')):
            highlighted_text = highlighted_text.replace(sentence, f'<span style="ba

    return highlighted_text

# Define the text
txt = """American football evolved in the United States, originating from the sport
The first American football match was played on November 6, 1869, between two colle
using rules based on the rules of soccer at the time.

A set of rule changes drawn up from 1880 onward by Walter Camp,
the "Father of American Football", established the snap, the line of scrimmage,
eleven-player teams, and the concept of downs! Later rule changes legalized the for
created the neutral zone and specified the size and shape of the football.

The sport is closely related to Canadian football!
It evolved in parallel with and at the same time as the American game,
although its rules were developed independently from those of Camp.

Most of the features that distinguish American football from rugby and soccer are a
```

```
The two sports are considered the primary variants of gridiron football!

American football is the most popular sport in the United States!

The most popular forms of the game are professional and college football,
with the other major levels being high school and youth football.
As of 2012, nearly 1.1 million high school athletes and 70,000 college athletes pla

# Display highlighted text in Jupyter Notebook
HTML(highlight_sentences_no_full_stop(txt))
```

Out[141…

American football evolved in the United States, originating from the sports of soccer and rugby! The first American football match was played on November 6, 1869, between two college teams, Rutgers and Princeton, using rules based on the rules of soccer at the time. A set of rule changes drawn up from 1880 onward by Walter Camp, the "Father of American Football", established the snap, the line of scrimmage, eleven-player teams, and the concept of downs! Later rule changes legalized the forward pass, created the neutral zone and specified the size and shape of the football. The sport is closely related to Canadian football! It evolved in parallel with and at the same time as the American game, although its rules were developed independently from those of Camp. Most of the features that distinguish American football from rugby and soccer are also present in Canadian football. The two sports are considered the primary variants of gridiron football! American football is the most popular sport in the United States! The most popular forms of the game are professional and college football, with the other major levels being high school and youth football. As of 2012, nearly 1.1 million high school athletes and 70,000 college athletes play the sport in the US annually.

Now find out all special symbols (non-alphanumeric, non-whitespace characters).

In [144…

```
import re

# Define the text
txt = """American football evolved in the United States, originating from the sport
The first American football match was played on November 6, 1869, between two colle
using rules based on the rules of soccer at the time.

A set of rule changes drawn up from 1880 onward by Walter Camp,
the "Father of American Football", established the snap, the line of scrimmage,
eleven-player teams, and the concept of downs! Later rule changes legalized the for
created the neutral zone and specified the size and shape of the football.

The sport is closely related to Canadian football!
It evolved in parallel with and at the same time as the American game,
although its rules were developed independently from those of Camp.

Most of the features that distinguish American football from rugby and soccer are a
The two sports are considered the primary variants of gridiron football!

American football is the most popular sport in the United States!
```

```
The most popular forms of the game are professional and college football,
with the other major levels being high school and youth football.
As of 2012, nearly 1.1 million high school athletes and 70,000 college athletes pla

# Define the pattern to match special symbols
pattern = r'[^\w\s]'

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

# Remove duplicates and sort for better readability
special_symbols = sorted(set(special_symbols))

# Display the special symbols
print(special_symbols)
```

```
['!', '"', ',', '-', '.']
```

Using the findall() method, check if the following text has any of the letters (a, r, n).

In [152…

```
import re

txt = "Hello, I am studyng Deep Learning at Mt. SAC. How are you doing?"

#Check if the string has any a, r, or n characters:

# Define the pattern to find the letters a, r, or n
pattern = r'[arn]'

# Use findall() to find all occurrences of the pattern
x = re.findall(pattern, txt)

# Print the result
print(x)
if x:
    print("Yes, there is at least one match!")
else:
    print("No match")
```

```
['a', 'n', 'a', 'r', 'n', 'n', 'a', 'a', 'r', 'n']
Yes, there is at least one match!
```

Other Examples

Search for a sequence that starts with "he", followed by two (any) characters, and an "o":

In [156…

```
import re

txt = "hello planet"

# Define the pattern to match the sequence
pattern = r'he..o'
```

```
# Use findall() to find all occurrences of the pattern
matches = re.findall(pattern, txt)

# Print the result
print(matches)
```

```
['hello']
```

Check if the string ends with "SAC":

In [158…

```
import re

txt = "Hello, I am studyng Deep Learning at Mt. SAC"

# Define the pattern to match "SAC" at the end of the string
pattern = r'SAC$'

# Use re.findall() to find all occurrences of the pattern
x = re.findall(pattern, txt)

# Print the result
print(x)

if x:
    print("Yes, there is a match!")
else:
    print("No match")
```

```
['SAC']
Yes, there is a match!
```

enter your full name below

In [160…

```
txt = "William Yeh"
# example:
# txt = "Angel Hernandez"

# Print the result
print(txt)
```

```
William Yeh
```

Using Named Group, extract your name and yoru last name.

Create a pattern, match, and use match.group()

In [163…

```
import re

# Define the text with your full name
txt = "William Yeh"

# Define the pattern with named groups
pattern = r'(?P<first_name>\w+) (?P<last_name>\w+)'
```

```
# Use re.match() to match the pattern
match = re.match(pattern, txt)

# Extract and print the named groups
if match:
    first_name = match.group('first_name')
    last_name = match.group('last_name')
    print(f"First Name: {first_name}")
    print(f"Last Name: {last_name}")
else:
    print("No match found")
```

```
First Name: William
Last Name: Yeh
```

In [33]:

In [ ]:

In [ ]:

Display your last name followed by your first name

In [165…

```
import re

# Define the text with your full name
txt = "William Yeh"

# Define the pattern with named groups
pattern = r'(?P<first_name>\w+) (?P<last_name>\w+)'

# Use re.match() to match the pattern
match = re.match(pattern, txt)

# Extract and print the named groups in reverse order
if match:
    first_name = match.group('first_name')
    last_name = match.group('last_name')
    print(f"Last Name: {last_name}")
    print(f"First Name: {first_name}")
else:
    print("No match found")
```

```
Last Name: Yeh
First Name: William
```

Explain why we need regular expression

Regular expressions (regex) are crucial for text processing due to their ability to define and search for complex patterns within strings. They are essential for validating input formats, such as email addresses or phone numbers, ensuring data meets specific criteria. Regex enables efficient data extraction, allowing you to pull out specific information from larger text bodies. Additionally, they support search and replace operations, making it easy to clean or format text. Regex can perform sophisticated text transformations and operations in a single line of code, enhancing efficiency. Their flexibility with pattern definitions makes them adaptable to various text processing needs. Integrated into many programming languages and tools, regex provides a universal solution for handling text data across different platforms.

In [ ]: