Module 7: Introduction to Regular Expressions

Regular expressions, often referred to as regex or regexp, are sequences of characters that define a search pattern. They are powerful tools for pattern matching and text manipulation in Python and other programming languages. Regular expressions allow you to specify complex search criteria, making it easier to extract, replace, or manipulate text based on specific patterns.

Basics of Regular Expressions

Literal Characters: Literal characters in a regular expression match themselves in the input string. For example, the regex `hello` will match the string "hello" in the input text.

Metacharacters: Metacharacters are special characters that have a reserved meaning in regular expressions. Some common metacharacters include:

- `.` (dot): Matches any single character except newline.
- `^` (caret): Matches the start of the string.
- `$` (dollar): Matches the end of the string.
- `*`, `+`, `?`: Quantifiers that specify the number of repetitions of the preceding character or group.
- `[]`: Character class, matches any one of the characters inside the brackets.
- `|` (pipe): Alternation, matches either the expression before or after the pipe.

Quantifiers: Quantifiers specify the number of occurrences of a character or group in a regular expression.

- `*`: Matches zero or more occurrences.
- `+`: Matches one or more occurrences.
- `?`: Matches zero or one occurrence.

Character Classes: Character classes allow you to match any one of a set of characters. For example, `[aeiou]` matches any vowel.

Examples

Matching a Specific Word:

```
import re

text = "Hello, world!"
pattern = r'Hello'
match = re.search(pattern, text)
if match:
 print("Match found:", match.group())
else:
 print("No match found.")
```

Matching Any Single Character:

```
import re

text = "cat, hat, bat"
pattern = r'.at'
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Matching Start and End of String:

```
import re

text = "Python is fun"
pattern_start = r'^Python'
pattern_end = r'fun$'

match_start = re.search(pattern_start, text)
match_end = re.search(pattern_end, text)

if match_start:
 print("Start of string matched:", match_start.group())

if match_end:
 print("End of string matched:", match_end.group())
```

Matching Repetitions:

```
import re
```

```
text = "aaaaaaabbbbccccc"
pattern = r'a+'
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Matching Character Classes:

```
import re

text = "apple, orange, banana"
pattern = r'[aeiou]'
matches = re.findall(pattern, text)
print("Vowels:", matches)
```

Regular expressions provide a flexible and efficient way to search, extract, and manipulate text based on specific patterns. By understanding the basics of regular expressions and using them effectively, you can perform complex text processing tasks with ease.

Exercises and Answers for Introduction to Regular Expressions

Literal Character Match:

Write a Python program that searches for the word "apple" in a given text string using a regular expression.

Single Character Match:

Create a regular expression pattern to match any three-letter word in a given text string. Test it with the string "cat dog hat bat".

Start and End Match:

Write a Python program to check if a given string starts with "Hello" and ends with "World" using regular expressions.

Repetitions Match:

Create a regular expression pattern to match sequences of consecutive digits in a given text string. Test it with the string "123abc456def789ghi".

Character Class Match:

Write a Python program that extracts all the vowels from a given text string using a regular expression.

Note: You can use the `re.search()` function to search for a pattern in a text string, and `re.findall()` function to find all occurrences of a pattern in a text string.

Answers

Literal Character Match:

```
import re

text = "I like apples."
pattern = r'apple'
match = re.search(pattern, text)
if match:
 print("Match found:", match.group())
else:
 print("No match found.")
```

Single Character Match:

```
import re

text = "cat dog hat bat"
pattern = r'\b\w{3}\b'
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Start and End Match:

```
import re
```

```
text = "Hello, World!"
pattern_start = r'^Hello'
pattern_end = r'World$'

match_start = re.search(pattern_start, text)
match_end = re.search(pattern_end, text)

if match_start:
 print("Start of string matched:", match_start.group())

if match_end:
 print("End of string matched:", match_end.group())
```

Repetitions Match:

```
import re

text = "123abc456def789ghi"
pattern = r'\d+'
matches = re.findall(pattern, text)
print("Matches:", matches)
```

Character Class Match:

```
import re

text = "apple, orange, banana"
pattern = r'[aeiou]'
matches = re.findall(pattern, text)
print("Vowels:", matches)
```

These exercises will help you practice using regular expressions to perform pattern matching tasks in Python. Regular expressions are powerful tools for text processing and can be used in various applications to search, extract, and manipulate text based on specific patterns.