Regex Tutorial

The term **Regex** stands for **Regular expression**. The **regex** or **regexp** or **regular expression** is a sequence of different characters which describe the particular search pattern. It is also referred/called as a **Rational expression**.
It is mainly used for searching and manipulating text strings. In simple words, you can easily search the pattern and replace them with the matching pattern with the help of regular expression.

This concept or tool is used in almost all the programming or scripting languages such as PHP, C, C++, Java, Perl, JavaScript, Python, Ruby, and many others. It is also used in word processors such as word which helps users for searching the text in a document, and also used in various IDEs.
The pattern defined by the regular expression is applied to the given string or a text from left to right.

Regular Expression Characters

There are following different type of characters of a regular expression:

1. Metacharacters
2. Quantifier
3. Groups and Ranges
4. Escape Characters or character classes

Metacharacters

Metacharacters	Description	Example
^	This character is used to match an expression to its right at the start of a string.	**^a** is an expression match to the string which starts with '**a**' such as "**aab", "a9c", "apr", "aaaaab"**, etc.
$	The $sign is used to match an expression to its left at the end of a string.	**r$** is an expression match to a string which ends with r such as "**aaabr", "ar", "r", "aannn9r"**, etc.
.	This character is used to match any single character in a string except the line terminator, i.e. /n.	**b.x** is an expression that match strings such as **"bax", "b9x", "bar"**.
|	It is used to match a particular character or a group of characters on either side. If the character on the left side is matched, then the right side's character is ignored.	**A|b** is an expression which gives various strings, but each string contains either **a** or **b**.
****	It is used to escape a special character after this sign in a string.	
A	It is used to match the character 'A' in the string.	This expression matches those strings in which at least one-time **A** is present. Such strings are "**Amcx", "mnAr", "mnopAx4"**.
Ab	It is used to match the substring 'ab' in the string.	This expression matches those strings in which **'Ab'** is present at least one time. Such strings are "**Abcx", "mnAb", "mnopAbx4"**.

Quantifiers

The quantifiers are used in the regular expression for specifying the number of occurrences of a character.

Characters	Description	Example
+	This character specifies an expression to its left for one or more times.	**s+** is an expression which gives **"s", "ss",** "**sss**", and so on.
?	This character specifies an expression to its left for 0 (Zero) or 1 (one)times.	**aS?** is an expression which gives either "**a**" or "**as", but not "ass".**
*	This character specifies an expression to its left for 0 or more times	**Br*** is an expression which gives **"B", "Br", "Brr", "Brrr"**, and so on…
{x}	It specifies an expression to its left for only x times.	**Mab{5}** is an expression which gives the following string which contains 5 b's: **"Mabbbbb"**
{x, }	It specifies an expression to its left for x or more times.	**Xb{3, }** is an expression which gives various strings containing at least 3 b's. Such strings are "**Xbbb", "Xbbbb"**, and so on.
{x,y}	It specifies an expression to its left, at least x times but less than y times.	**Pr{3,6}a** is an expression which provides two strings. Both strings are as follows: **"Prrrr" and "Prrrrr"**

Groups and Ranges

The groups and ranges in the regular expression define the collection of characters enclosed in the brackets.

Characters	Description	Example
( )	It is used to match everything which is in the simple bracket.	**A(xy)** is an expression which matches with the following string: **"Axy"**
{ }	It is used to match a particular number of occurrences defined in the curly bracket for its left string.	**xz{4,6}** is an expression which matches with the following string: **"xzzzzz"**
[ ]	It is used to match any character from a range of characters defined in the square bracket.	**xz[atp]r** is an expression which matches with the following strings: **"xzar", "xztr", and "xzpr"**
[pqr]	It matches p, q, or r individually.	Following strings are matched with this expression: **"p", "q", and "r".**
[pqr][xy]	It matches p, q, or r, followed by either x or y.	Following strings are matched with this expression: **"px", "qx", and "rx", "py", "qy", and "ry".**
(?: …)	It is used for matching a non-capturing group.	A(?:nt|pple) is an expression which matches to the following string: **"Apple"**
[^…..]	It matches a character which is not defined in the square bracket.	Suppose, **Ab[^pqr]** is an expression which matches only the following string: **"Ab"**
[a-z]	It matches letters of a small case from a to z.	This expression matches the strings such as: **"a", "python", "good".**
[A-Z]	It matches letters of an upper case from A to Z.	This expression matches the strings such as: **"EXCELLENT", "NATURE".**

^[a-zA-Z]	It is used to match the string, which is either starts with a small case or upper-case letter.	This expression matches the strings such as: **"A854xb", "pv4fv", "cdux".**
[0-9]	It matches a digit from 0 to 9.	This expression matches the strings such as: **"9845", "54455"**
[aeiou]	This square bracket only matches the small case vowels.	-
[AEIOU]	This square bracket only matches the upper-case vowels.	-
ab[^4-9]	It matches those digits or characters which are not defined in the square bracket.	This expression matches those strings which do not contain 5, 6, 7, and 8.

Escape Characters or Character Classes

Characters	Description
\s	It is used to match a one white space character.
\S	It is used to match one non-white space character.
\0	It is used to match a NULL character.
\a	It is used to match a bell or alarm.
\d	It is used to match one decimal digit, which means from 0 to 9.
\D	It is used to match any non-decimal digit.
\n It helps a user to match a new line.	
\w	It is used to match the alphanumeric [0-9a-zA-Z] characters.
\W	It is used to match one non-word character
\b	It is used to match a word boundary.

Regular Expression in Different Languages

There are following scripting and programming languages which use the regular expression:

1. Use of Regular Expression in Java
2. Use of Regular Expression in PHP
3. Use of Regular Expression in Python
4. Use of Regular Expression in JavaScript

Use of Regular Expression in Java (Java Regex)

In Java language, **Regex** or **Regular Expression** is an application programming interface which is used for manipulating, searching, and editing a string. You can use the regular expression in java by importing the java.util.regex API package in your code.

There are the following three classes which comes under the **java.util.regex** package:

1. **regex.Pattern:** This class helps in defining the patterns
2. **regex.Matcher:** This class helps in performing the match operations on an inputted string using patterns.
3. **PatternSyntaxException:** This class helps the users by indicating the syntax error in a regular expression pattern.

util.regex.Pattern Class

This class (util.regex.Pattern)is a compiled version of Regex and can be called by the compile() method. The compile() method accepts the regex as a first argument. This class does not provide any public constructor.

Following are the different functions in Pattern class:

Methods()	Description
static Pattern compile(String regex)	This method is used to compile the given regex into a pattern.
Matcher Matcher (CharSequence input)	This method creates a matcher for matching the pattern with the given string, which is inputted by a user.
String toString()	This method returns the representation of the string.
String[] split(CharSequence input)	This method splits a string on the basis of matches.
Static Boolean matches(String regex CharSequence input)	This method matches the regular expression against the given input string.

util.regex.Matcher Class

This class is used for performing the match operation on the input string by calling a matcher() function on any object or Pattern. This class does not define any public constructor.

Following are the different functions in Matcher class:

Methods()	Description
boolean matches()	This method checks whether the regular expression matches the pattern or not.
boolean find()	This method searches the occurrences of regex in a string.
boolean find(int start)	This method searches the occurrences of regex in a string from the starting index.
String group()	This method helps in finding the matched subsequence.
int start()	This method helps in returning the starting index of the matched subsequence.
int end()	This method helps in returning the last index of the matched subsequence.
int groupCount()	This method helps in returning the total number of the matched subsequence.

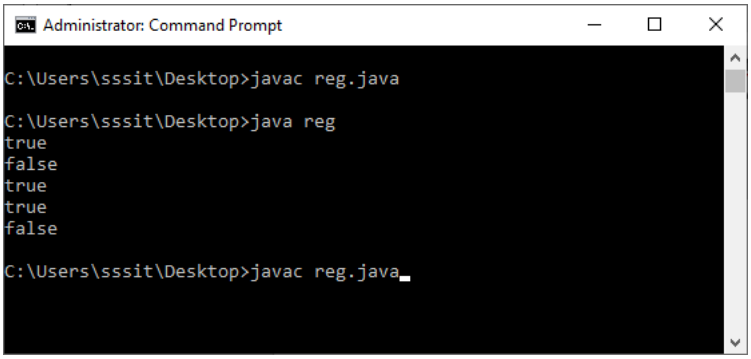
Examples of use Regular Expression in Java

Example 1: This example helps in understanding the dot operator in Java code.

```
import java.util.regex.*;
class reg{
public static void main(String args[]){
System.out.println(Pattern.matches(".r.", "arp")); // This statement displays Boolean value True because the second character is r in both st
System.out.println(Pattern.matches(".bm", "abc")); //This statement displays Boolean value False because the third character is different in
System.out.println(Pattern.matches("..m", "msm")); //This statement displays Boolean value True because the third character is m in both s
System.out.println(Pattern.matches("a..s", "amns")); //This statement displays Boolean value True because the first and last character is sa
System.out.println(Pattern.matches(".s.", "mas")); //This statement displays Boolean value False because the second character is different i
}
}
```

Test it Now

Output:



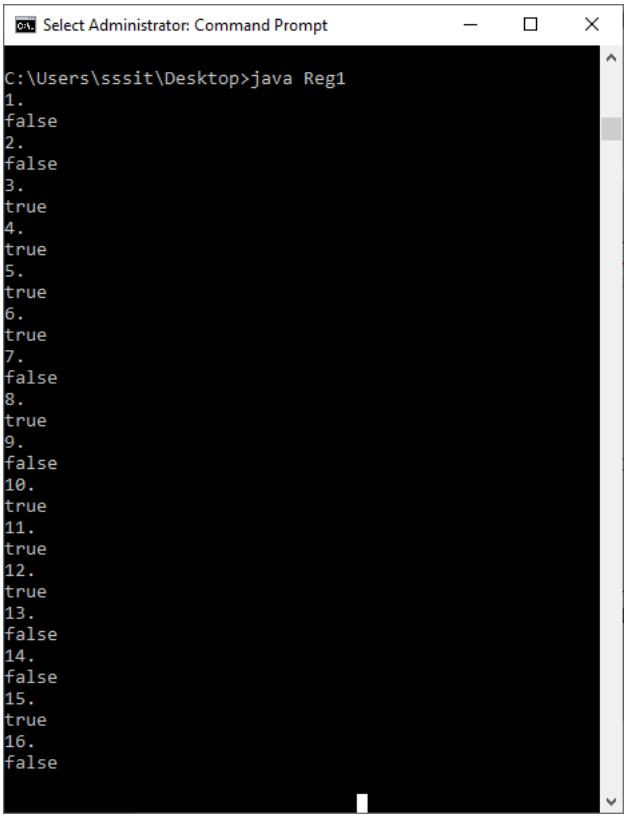
Example 2: The following example uses the different characters of regular expression.

```
import java.util.regex.*;
class Reg1{
public static void main(String args[]){

System.out.println(" 1. ");
System.out.println(
        Pattern.matches("^[ a-z]om", "Tom")); // This statement returns False because the string starts with the Upper-
case letter which does not match with the regex.

System.out.println(" 2. ");
System.out.println(
                                                                    Pattern.matches("
[Rpq]om", "Tom")); // This statement returns False because the string starts with 'T', which does not match with any character R, P, or Q of P

System.out.println(" 3. ");
System.out.println(
                                                                    Pattern.matches("
[Tt]om", "Tom")); // This statement returns True because the string starts with 'T' which matches with a character T from Regex.

System.out.println(" 4. ");
System.out.println(
   Pattern.matches("Cat|Rat", "Rat")); // This statement returns True because the string matches with the Second part from the Regular Exp

System.out.println(" 5. ");
System.out.println(
                                                                Pattern.matches("[CM]at|
[Bb]ad", "Bad")); // This statement returns True because the string matches with the Second part from the Regular Expression.

System.out.println(" 6. ");
System.out.println(
   Pattern.matches(".*bit.*", "rabbit")); // This statement returns True because the string contains the bit which match to the Regular Expre

System.out.println(" 7. ");
System.out.println(
   Pattern.matches("^[\\d].*", "abc")); // This statement returns True because the string starts with the letter not a digit, which does not ma

System.out.println(" 8. ");
System.out.println(
   Pattern.matches("^[^\\d].*", "abc123")); // This statement returns True because the string starts with the letter, which matches with the

System.out.println(" 9. ");
  System.out.println(
```

```
                                                    Pattern.matches("[a-zA-Z][a-zA-Z][0-9A-
Z]", "aPz")); // This statement returns False because the last character 'z' in string does not match with the  with the '0-9A-
Z' in regular expression.

System.out.println(" 10. ");
  System.out.println(
                                                    Pattern.matches("[a-zA-Z][a-zA-Z][a-zA-
Z]", "aAA")); // This statement returns True because the all the characters in string matches with the  with the regular expression.

System.out.println(" 11. ");
System.out.println(
    Pattern.matches("java[tT]poin[tT]$", "javaTpoint")); // This statement returns True because the string ends with the 't' character which m

System.out.println(" 12. ");
System.out.println(
    Pattern.matches("\\D*", "abcde")); // This statement returns True because the string does not contain any digit, so it follows the regular

System.out.println(" 13. ");
System.out.println(
    Pattern.matches("\\D*", "abcde123")); // This statement returns False because the string contains the digits, so it does not follow the reg
}
}
```

Test it Now

Output:



Use of Regular Expression in PHP

You can use the regular expression (regex) in the PHP by taking the help of functions of PCRE (Perl Compatible Regular Expression) library.

There are different types of built-in functions in the following table, which are used for working with regular expressions. These functions are case-sensitive.

Functions	Description
preg_match()	The **preg_match()** function returns true if a regular expression matches a specific string, otherwise false.
preg_match_all()	The **preg_match_all()** function is used for matching all the occurrences of pattern in a string.
preg_replace()	The **preg_replace()** function matches a pattern and then replace it with the string.
preg_split()	The **preg_split()** function divides the string into substring using a regex.
Preg_grep()	The **preg_grep()** function is used to return all those elements of input array which matches the regular expression pattern.

Examples of use Regular Expression in PHP

Example 1: The following example helps in understanding how to use the character class or a group in a regular expression for matching with string.

```
<?php
$regExp = "/[a-zA-Z]+ \d+/";
if (preg_match($regExp, "Januaury 26")) // This statement matches the regular expression with the string, If mathces then if statement exe
{
   echo "The regular expression and string are the same.";
} else
{
   echo "The regex pattern does not match with the string.";
}

?>
```

The above program of php gives the following output:

```
The regular expression and string are the same.
```

Example 2: The following example helps in understanding how to use the '^' character in the PHP code.

```
<?php
$regexp = "/^J/";
$employeenames = array("Clark Kent","John Carter", "John Rambo");
$matches = preg_grep($regexp, $employeenames); // This statement store those names in $matches variables which are start with the Upp
case J letter.
// Loop through matches array and display matched names
foreach($matches as $match){
   echo $match . "<br>";
}
?>
```

The above program of php gives the following output:

```
John Carter
John Rambo
```

Use of Regular Expression in Python (Python Regex)

You can use the regular expression (Regex) in the code of Python by importing the re module in your script. This module defines the various function or methods which are used for handling the regular expression.

The following table defines the various functions:

Methods	Description
re.match()	The **re.match()** method is used to return a string which is matched with the regular expression.
re.search()	The **re.search()** method returns an object of the match when the pattern is found in a string or text.
re.findall()	The **re.findall()** method is used to return a string list containing all the matches.
re.split()	The **re.split()** method is used to divide the string on the basis of matching with the regular expression.
re.sub()	The **re.sub()** method is used to replace the matched string with another string.

Examples of use Regular Expression in Python

Example 1: This example helps in understanding how to use the **findall()** method in python script.

```
import re

string = 'Fruits 32, Animals 80, Cars 34'
pattern = '\D+'

match = re.findall(pattern, string)      #This statement is used to store the matching values on the basis of a given pattern from the string.
print(match)   #This statement displays the values which are stored in match variable
```

The above program of python with regular expression gives the following output:

```
['Fruits ', ', Animals ', ', Cars ']
```

Example 2: This example helps in understanding how to use the **split(), search(), and sub()** methods in python script.

```
import re
string = 'Zero:0 one:1 Two:2 Three:3 Four:4 Five:5 Six:6 Seven:7 eight:8 Nine:9 Ten:10 Twenty:20 Thirty:30 Forty:40 Fifty:50 Sixty:60 Seventy
regex = '\d+'

splitval = re.split(regex, string)     #This statement splits the string on the basis of matching values between pattern and string.
print(splitval)

string = 'a1 \nb2 \nc4'
pattern = '\d' # This statement defines the regular expression for matching with the string.

# empty string
replace = 's'
new_string = re.sub(pattern, replace, string)      # This statement replaces those matched characters with a string stored in a replace variab
print(new_string)    # This statement displays the new string after the replacement of characters.

text = "Regular Expression is also referred as Regex."
regex= "\d"
res = re.search(regex, text)   # This statement searches the regular expression in a string.
if res:
  print("Regular expression found inside the string")
```

```
else:
 print("Regular expression not found inside the String")
```

The above program of python with regular expression gives the following output:

```
['Zero:', ' one:', ' Two:', ' Three:', ' Four:', ' Five:', ' Six:', ' Seven:', ' eight:', ' Nine:', ' Ten:', ' Twenty:', ' Thirty:', ' Forty:', ' Fifty:', ' Sixty:', ' Seventy:', ' Eig
```

```
as
bs
cs
```

```
pattern not found inside the String
```

Use of Regular Expression in JavaScript

You can easily use the regular expression in the JavaScript code by the help of following two string methods:

1. **search():** This method searches the regular expression in the string and also returns the position where the match found.
2. **Replace():** This method is used to return the string after the replacement of a matched character in a string.

Examples of Regular Expression in JavaScript

Example 1: This example uses the **search()** method in the JavaScript script for understanding the regular expression.

```
<!DOCTYPE html>
<html>
<head>
<script>
var string = "Our Site is helpfull for studying about technical courses.!";
pattern="technical";
var res = string.search(pattern); /* This statement stores the position of the pattern in a string, if it is found in a string. */
document.write("Position of the pattern in a string:");
document.write(res);
</script>
</head>
<body>
</body>
</html>
```

Test it Now

The above program of JavaScript with regular expression gives the following output:

```
Position of the pattern in a string:40
```

Example 2: This example uses the **replace()** method in the JavaScript script for understanding the regular expression.

```
<html>
<head>
<script>
var string = "You are a Bad Student";
var pattern=/Bad/;
var replace="Good";
var res = string.replace(/Bad/,replace);
```

```
/* The above statement replaces the Bad word from the string by the Good word using the replace method. */
document.write("After replacing the substring, the modified string is:"+ '<br>');
document.write(res);
</script>
</head>
<body>
</body>
</html>
```

Test it Now

The above program of JavaScript with regular expression gives the following output:

```
After replacing the substring, the modified string is:
You are a Good Student
```

Company Questions

Trending Technologies

- Artificial Intelligence — Artificial Intelligence
- AWS Tutorial — AWS
- Selenium tutorial — Selenium
- Cloud Computing — Cloud Computing
- Hadoop tutorial — Hadoop
- ReactJS Tutorial — ReactJS
- Data Science Tutorial — Data Science
- Angular 7 Tutorial — Angular 7
- Blockchain Tutorial — Blockchain
- Git Tutorial — Git
- Machine Learning Tutorial — Machine Learning
- DevOps Tutorial — DevOps

B.Tech / MCA

- DBMS tutorial — DBMS
- Data Structures tutorial — Data Structures
- DAA tutorial — DAA
- Operating System — Operating System
- Computer Network tutorial — Computer Network
- Compiler Design tutorial — Compiler Design
- Computer Organization and Architecture — Computer Organization
- Discrete Mathematics Tutorial — Discrete Mathematics
- Ethical Hacking — Ethical Hacking
- Computer Graphics Tutorial — Computer Graphics
- Software Engineering — Software Engineering
- html tutorial — Web Technology
- Cyber Security tutorial — Cyber Security
- Automata Tutorial — Automata
- C Language tutorial — C Programming
- C++ tutorial — C++
- Java tutorial — Java
- .Net Framework tutorial — .Net
- Python tutorial — Python
- List of Programs — Programs
- Control Systems tutorial — Control System
- Data Mining Tutorial — Data Mining
- Data Warehouse Tutorial — Data Warehouse