To ensure that it is easy to decipher, I’ll look at this one character by character, too:

* ^ (caret) at the beginning of the regular expression, or following the vertical bar (|), means that the phone number will be at the beginning of a line.
* ( opens a capturing group.
* \( is a literal open parenthesis.
* \d matches a digit.
* {3} is a quantifier that, following \d, matches exactly three digits.
* \) matches a literal close parenthesis.
* | (the vertical bar) indicates alternation, that is, a given choice of alternatives. In other words, this says “match an area code with parentheses or without them.”
* ^ matches the beginning of a line.
* \d matches a digit.
* {3} is a quantifier that matches exactly three digits.
* [.-]? matches an optional dot or hyphen.
* ) close capturing group.
* ? make the group optional, that is, the prefix in the group is not required.
* \d matches a digit.
* {3} matches exactly three digits.
* [.-]? matches another optional dot or hyphen.
* \d matches a digit.
* {4} matches exactly four digits.
* $ matches the end of a line.

#### **Character Classes**

People also askWhat is a regular expression?What is a regular expression?A regular expression is a method used in programming for pattern matching. Regular expressions provide a flexible and concise means to match strings of text. For example, a regular expression could be used to search through large volumes of text and change all occurrences of "cat" to "dog".What is a Regular Expression? - Definition from Techopedia

Enter your regex: [bcr]at

Enter input string to search: bat

I found the text "bat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: cat

I found the text "cat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: rat

I found the text "rat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: hat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: bat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: cat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: rat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: hat

I found the text "hat" starting at index 0 and ending at index 3.

Enter your regex: [a-c]

Enter input string to search: a

I found the text "a" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: b

I found the text "b" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: c

I found the text "c" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: d

No match found.

Enter your regex: foo[1-5]

Enter input string to search: foo1

I found the text "foo1" starting at index 0 and ending at index 4.

Enter your regex: foo[1-5]

Enter input string to search: foo5

I found the text "foo5" starting at index 0 and ending at index 4.

Enter your regex: foo[1-5]

Enter input string to search: foo6

No match found.

Enter your regex: foo[^1-5]

Enter input string to search: foo1

No match found.

Enter your regex: foo[^1-5]

Enter input string to search: foo6

I found the text "foo6" starting at index 0 and ending at index 4.

Enter your regex: [0-4[6-8]]

Enter input string to search: 0

I found the text "0" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 5

No match found.

Enter your regex: [0-4[6-8]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 8

I found the text "8" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 9

No match found.

Enter your regex: [0-9&&[345]]

Enter input string to search: 3

I found the text "3" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 4

I found the text "4" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 5

I found the text "5" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 2

No match found.

Enter your regex: [0-9&&[345]]

Enter input string to search: 6

No match found.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 3

No match found.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 4

I found the text "4" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 5

I found the text "5" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 7

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 2

I found the text "2" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 3

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 4

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 5

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 9

I found the text "9" starting at index 0 and ending at index 1.

#### **Java - Regular Expressions**

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. They can be used to search, edit, or manipulate text and data. The java.util.regex package primarily consists of the …

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMatches {

public static void main( String args[] ) {

// String to be scanned to find the pattern.

String line = "This order was placed for QT3000! OK?";

String pattern = "(.\*)(\\d+)(.\*)";

// Create a Pattern object

Pattern r = Pattern.compile(pattern);

// Now create matcher object.

Matcher m = r.matcher(line);

if (m.find( )) {

System.out.println("Found value: " + m.group(0) );

System.out.println("Found value: " + m.group(1) );

System.out.println("Found value: " + m.group(2) );

}else {

System.out.println("NO MATCH");

}

}

}

Found value: This order was placed for QT3000! OK?

Found value: This order was placed for QT300

Found value: 0

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMatches {

private static final String REGEX = "\\bcat\\b";

private static final String INPUT = "cat cat cat cattie cat";

public static void main( String args[] ) {

Pattern p = Pattern.compile(REGEX);

Matcher m = p.matcher(INPUT); // get a matcher object

int count = 0;

while(m.find()) {

count++;

System.out.println("Match number "+count);

System.out.println("start(): "+m.start());

System.out.println("end(): "+m.end());

}

}

}

Match number 1

start(): 0

end(): 3

Match number 2

start(): 4

end(): 7

Match number 3

start(): 8

end(): 11

Match number 4

start(): 19

end(): 22

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMatches {

private static final String REGEX = "foo";

private static final String INPUT = "fooooooooooooooooo";

private static Pattern pattern;

private static Matcher matcher;

public static void main( String args[] ) {

pattern = Pattern.compile(REGEX);

matcher = pattern.matcher(INPUT);

System.out.println("Current REGEX is: "+REGEX);

System.out.println("Current INPUT is: "+INPUT);

System.out.println("lookingAt(): "+matcher.lookingAt());

System.out.println("matches(): "+matcher.matches());

}

}

Current REGEX is: foo

Current INPUT is: fooooooooooooooooo

lookingAt(): true

matches(): false

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMatches {

private static String REGEX = "dog";

private static String INPUT = "The dog says meow. " + "All dogs say meow.";

private static String REPLACE = "cat";

public static void main(String[] args) {

Pattern p = Pattern.compile(REGEX);

// get a matcher object

Matcher m = p.matcher(INPUT);

INPUT = m.replaceAll(REPLACE);

System.out.println(INPUT);

}

}

The cat says meow. All cats say meow.

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMatches {

private static String REGEX = "a\*b";

private static String INPUT = "aabfooaabfooabfoob";

private static String REPLACE = "-";

public static void main(String[] args) {

Pattern p = Pattern.compile(REGEX);

// get a matcher object

Matcher m = p.matcher(INPUT);

StringBuffer sb = new StringBuffer();

while(m.find()) {

m.appendReplacement(sb, REPLACE);

}

m.appendTail(sb);

System.out.println(sb.toString());

}

}

-foo-foo-foo-

#### **Java Guides**

This Java example demonstrates how to write a regular expression to validate user input in such a way that it allows only alphanumeric characters. Alphanumeric characters are all alphabets and numbers i.e. letters A–Z, a–z, and digits 0–9.

package net.javaguides.corejava.regex;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexExample {

public static void main(String[] args) {

List < String > words = Arrays.asList("One", "Two",

"Three", "Four", "Five", "Six", "Seven", "Maven", "Amen", "eleven");

Pattern p = Pattern.compile(".even");

for (String word: words) {

Matcher m = p.matcher(word);

if (m.matches()) {

System.out.printf("%s -> matches%n", word);

} else {

System.out.printf("%s -> does not match%n", word);

}

}

}

}

One -> does not match

Two -> does not match

Three -> does not match

Four -> does not match

Five -> does not match

Six -> does not match

Seven -> matches

Maven -> does not match

Amen -> does not match

eleven -> does not match

package net.javaguides.corejava.regex;

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaAlphanumericRegex {

public static void main(String[] args) {

List < String > names = new ArrayList < String > ();

names.add("JavaGuides");

names.add("JavaGuides123");

names.add("JavaGuides-----////"); //Incorrect

String regex = "^[a-zA-Z0-9]+$";

Pattern pattern = Pattern.compile(regex);

for (String name: names) {

Matcher matcher = pattern.matcher(name);

System.out.println(matcher.matches());

}

}

}

true

true

false

package net.javaguides.corejava.regex;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexAnchorExample {

public static void main(String[] args) {

List < String > sentences = Arrays.asList("I am looking for Prabhas",

"Prabhas is an Actor",

"Mahesh and Prabhas are close friends");

Pattern p = Pattern.compile("^Prabhas");

for (String word: sentences) {

Matcher m = p.matcher(word);

if (m.find()) {

System.out.printf("%s -> matches%n", word);

} else {

System.out.printf("%s -> does not match%n", word);

}

}

}

}

I am looking for Prabhas -> does not match

Prabhas is an Actor -> matches

Mahesh and Prabhas are close friends -> does not match

package net.javaguides.corejava.regex;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexAlternation {

public static void main(String[] args) {

List < String > users = Arrays.asList("Ramesh", "Tom", "Tony",

"Rocky", "John", "Prabhas");

Pattern p = Pattern.compile("Ramesh|Tom|Prabhas|Rocky");

for (String user: users) {

Matcher m = p.matcher(user);

if (m.matches()) {

System.out.printf("%s -> matches%n", user);

} else {

System.out.printf("%s -> does not match%n", user);

}

}

}

}

Ramesh -> matches

Tom -> matches

Tony -> does not match

Rocky -> matches

John -> does not match

Prabhas -> matches

package net.javaguides.corejava.regex;

public class CheckPhoneExample {

public static void main(String[] args) {

System.out.println("Phone number 1234567890 validation result: " + validatePhoneNumber("1234567890"));

System.out.println("Phone number 123-456-7890 validation result: " + validatePhoneNumber("123-456-7890"));

System.out.println(

"Phone number 123-456-7890 x1234 validation result: " + validatePhoneNumber("123-456-7890 x1234"));

System.out.println(

"Phone number 123-456-7890 ext1234 validation result: " + validatePhoneNumber("123-456-7890 ext1234"));

System.out.println("Phone number (123)-456-7890 validation result: " + validatePhoneNumber("(123)-456-7890"));

System.out.println("Phone number 123.456.7890 validation result: " + validatePhoneNumber("123.456.7890"));

System.out.println("Phone number 123 456 7890 validation result: " + validatePhoneNumber("123 456 7890"));

}

private static boolean validatePhoneNumber(String phoneNo) {

// validate phone numbers of format "1234567890"

if (phoneNo.matches("\\d{10}"))

return true;

// validating phone number with -, . or spaces

else if (phoneNo.matches("\\d{3}[-\\.\\s]\\d{3}[-\\.\\s]\\d{4}"))

return true;

// validating phone number with extension length from 3 to 5

else if (phoneNo.matches("\\d{3}-\\d{3}-\\d{4}\\s(x|(ext))\\d{3,5}"))

return true;

// validating phone number where area code is in braces ()

else if (phoneNo.matches("\\(\\d{3}\\)-\\d{3}-\\d{4}"))

return true;

// return false if nothing matches the input

else

return false;

}

}

Phone number 1234567890 validation result: true

Phone number 123-456-7890 validation result: true

Phone number 123-456-7890 x1234 validation result: true

Phone number 123-456-7890 ext1234 validation result: true

Phone number (123)-456-7890 validation result: true

Phone number 123.456.7890 validation result: true

Phone number 123 456 7890 validation result: true

package net.javaguides.corejava.regex;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexCurrencySymbol {

public static void main(String[] args) {

String content = "Let's find the symbols or currencies: $ Dollar, € Euro, ¥ Yen";

String regex = "\\p{Sc}";

Pattern pattern = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(content);

while (matcher.find()) {

System.out.print("Start index: " + matcher.start());

System.out.print(" End index: " + matcher.end() + " ");

System.out.println(" : " + matcher.group());

}

}

}

Start index: 39 End index: 40 : $

Start index: 49 End index: 50 : €

Start index: 57 End index: 58 : ¥

package net.javaguides.corejava.regex;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexGroups {

public static void main(String[] args) {

String content = "<p>The <code>Pattern</code> is a compiled " +

"representation of a regular expression.</p>";

Pattern p = Pattern.compile("(</?[a-z]\*>)");

Matcher matcher = p.matcher(content);

while (matcher.find()) {

System.out.println(matcher.group(1));

}

}

}

<p>

<code>

</code>

</p>

package net.javaguides.corejava.regex;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexCaseInsensitive {

public static void main(String[] args) {

List < String > users = Arrays.asList("dog", "Dog", "DOG", "Doggy");

Pattern p = Pattern.compile("dog", Pattern.CASE\_INSENSITIVE);

users.forEach((user) - > {

Matcher m = p.matcher(user);

if (m.matches()) {

System.out.printf("%s matches%n", user);

} else {

System.out.printf("%s does not match%n", user);

}

});

}

}

dog matches

Dog matches

DOG matches

Doggy does not match

package net.javaguides.corejava.regex;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class JavaRegexEmail {

public static void main(String[] args) {

List < String > emails = Arrays.asList("ramesh@gmail.com",

"tom@yahoocom", "34234sdfa#2345", "tony@gmail.com");

String regex = "^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9-]+\\.[a-zA-Z.]{2,18}$";

Pattern p = Pattern.compile(regex);

for (String email: emails) {

Matcher m = p.matcher(email);

if (m.matches()) {

System.out.printf("%s matches%n", email);

} else {

System.out.printf("%s does not match%n", email);

}

}

}

}

ramesh@gmail.com matches

tom@yahoocom does not match

34234sdfa#2345 does not match

tony@gmail.com matches

package net.javaguides.corejava.regex;

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexMinMaxLength {

public static void main(String[] args) {

List < String > names = new ArrayList < String > ();

names.add("RAMESH");

names.add("JAVAGUIDES");

names.add("RAMESHJAVAGUIDES"); //Incorrect

names.add("RAMESH890"); //Incorrect

String regex = "^[A-Z]{1,10}$";

Pattern pattern = Pattern.compile(regex);

for (String name: names) {

Matcher matcher = pattern.matcher(name);

System.out.println(matcher.matches());

}

}

}

true

true

false

false

#### **Java Regular Expression (RegEx) Explained [Easy Examples]**

A Java regular expression is a special sequence of characters that helps us to match or find other strings or sets of strings, using a specified syntax held in a pattern. They can be used to …

// importing regex class

import java.util.regex.\*;

// java main class

public class Main{

// java main method

public static void main(String args[])

{

// printing match one:

System.out.println("Matching one is:");

//Using compile() matches() and matcher() methods

// return true

boolean match1=Pattern.compile("lin.x").matcher("linux").matches();

// printing out the matching one

System.out.println(match1);

// printing

System.out.println("Matching two is:");

// .. represents 2 characters

// also returns true

boolean match2 = Pattern.matches("clo..", "cloud");

// printing match two

System.out.println(match2);

// printing

System.out.println("Matching three is:");

// .. .represents 2 characters

// returns false

boolean match3 = Pattern.matches("golinux..", "golinuxcloud");

// printing match three

System.out.println(match3);

}

}

Matching one is:

true

Matching two is:

true

Matching three is:

false

// importing regex

import java.util.regex.\*;

// java main class

public class Main {

// java main method

public static void main(String args[]){

//Case Sensitive Searching

// Creating new pattern "linux" for search

Pattern pattern = Pattern.compile("Linux");

// Searching linux in the given pattern

Matcher match = pattern.matcher("GoLinuxcloud");

// Printing start and end indexes of the pattern in text

System.out.println("The following is the result of search:");

// using while loop

while (match.find())

// printing the pattern index

System.out.println("Pattern index starts from " + match.start() +

" to " + (match.end()-1));

}

}

Pattern index starts from 2 to 6

// importing regex

import java.util.regex.\*;

// java main class

public class Main {

// java main method

public static void main(String args[]){

//Case Insensitive Searching

// creating pattern of linux with lower case

Pattern pattern1= Pattern.compile("linux", Pattern.CASE\_INSENSITIVE);

// Searching linux the word linux in the GoLinuxcloud

Matcher match1 = pattern1.matcher("GoLinuxcloud");

// using while loop

while (match1.find())

// printing the index of search

System.out.println("Pattern starts from " + match1.start() +

" to " + (match1.end()-1));

}

}

Pattern starts from 2 to 6

// importing regex

import java.util.regex.\*;

// java main method

public class Main {

// creating some statice varibles of type private

private static String REGEX = "[";

private static String INPUT = "Welcome to " + "Golinuxcould.org";

private static String REPLACE = "com";

// java main method

public static void main(String[] args) {

// try block to handle error

try{

// creating pattern

Pattern pattern = Pattern.compile(REGEX);

// get a matcher object

Matcher matcher = pattern.matcher(INPUT);

// replacing

INPUT = matcher.replaceAll(REPLACE);

}

// catching the patternSyntaxException

catch(PatternSyntaxException e){

System.out.println("PatternSyntaxException: ");

System.out.println("Description: "+ e.getDescription());

System.out.println("Index: "+ e.getIndex());

System.out.println("Message: "+ e.getMessage());

System.out.println("Pattern: "+ e.getPattern());

}

}

}

PatternSyntaxException:

Description: Unclosed character class

Index: 0

Message: Unclosed character class near index 0

[

^

Pattern: [

#### **How to match "any character" in regular expression?**

If you are trying to match anything except whitespace you can try [\S] {min\_char\_to\_match,}. Try the regex . {3,}. This will match all characters except a new line. …

AAA123

ABCDEFGH123

XXXX123

Pattern pattern = Pattern.compile(".\*123", Pattern.DOTALL);

Matcher matcher = pattern.matcher(inputStr);

boolean matchFound = matcher.matches();

[\\s\\S]\*

[\\d\\D]\*

[\\w\\W]\*

[\\s\\S]+

[\\d\\D]+

[\\w\\W]+

[\s\S]\*

[\d\D]\*

[\w\W]\*

[\s\S]+

[\d\D]+

[\w\W]+

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegularExpression{

public static void main(String[] args){

final String regex\_1 = "[\\s\\S]\*";

final String regex\_2 = "[\\d\\D]\*";

final String regex\_3 = "[\\w\\W]\*";

final String string = "AAA123\n\t"

+ "ABCDEFGH123\n\t"

+ "XXXX123\n\t";

final Pattern pattern\_1 = Pattern.compile(regex\_1);

final Pattern pattern\_2 = Pattern.compile(regex\_2);

final Pattern pattern\_3 = Pattern.compile(regex\_3);

final Matcher matcher\_1 = pattern\_1.matcher(string);

final Matcher matcher\_2 = pattern\_2.matcher(string);

final Matcher matcher\_3 = pattern\_3.matcher(string);

if (matcher\_1.find()) {

System.out.println("Full Match for Expression 1: " + matcher\_1.group(0));

}

if (matcher\_2.find()) {

System.out.println("Full Match for Expression 2: " + matcher\_2.group(0));

}

if (matcher\_3.find()) {

System.out.println("Full Match for Expression 3: " + matcher\_3.group(0));

}

}

}

Full Match for Expression 1: AAA123

ABCDEFGH123

XXXX123

Full Match for Expression 2: AAA123

ABCDEFGH123

XXXX123

Full Match for Expression 3: AAA123

ABCDEFGH123

XXXX123

String[] tests = {

"AAA123",

"ABCDEFGH123",

"XXXX123",

"XYZ123ABC",

"123123",

"X123",

"123",

};

for (String test : tests) {

System.out.println(test + " " +test.matches(".+123"));

}

/a[^]\*Z/.test("abcxyz \0\r\n\t012789ABCXYZ") // Returns ‘true’.

String value = "|°¬<>!\"#$%&/()=?'\\¡¿/\*-+\_@[]^^{}";

String expression = "[a-zA-Z0-9\\p{all}]{0,50}";

if(value.matches(expression)){

System.out.println("true");

} else {

System.out.println("false");

}

#### **Character Classes**

This Java tutorial describes exceptions, basic input/output, concurrency, regular expressions, and the platform environment In the context of regular expressions, a character class is a …

Enter your regex: [bcr]at

Enter input string to search: bat

I found the text "bat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: cat

I found the text "cat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: rat

I found the text "rat" starting at index 0 and ending at index 3.

Enter your regex: [bcr]at

Enter input string to search: hat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: bat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: cat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: rat

No match found.

Enter your regex: [^bcr]at

Enter input string to search: hat

I found the text "hat" starting at index 0 and ending at index 3.

Enter your regex: [a-c]

Enter input string to search: a

I found the text "a" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: b

I found the text "b" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: c

I found the text "c" starting at index 0 and ending at index 1.

Enter your regex: [a-c]

Enter input string to search: d

No match found.

Enter your regex: foo[1-5]

Enter input string to search: foo1

I found the text "foo1" starting at index 0 and ending at index 4.

Enter your regex: foo[1-5]

Enter input string to search: foo5

I found the text "foo5" starting at index 0 and ending at index 4.

Enter your regex: foo[1-5]

Enter input string to search: foo6

No match found.

Enter your regex: foo[^1-5]

Enter input string to search: foo1

No match found.

Enter your regex: foo[^1-5]

Enter input string to search: foo6

I found the text "foo6" starting at index 0 and ending at index 4.

Enter your regex: [0-4[6-8]]

Enter input string to search: 0

I found the text "0" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 5

No match found.

Enter your regex: [0-4[6-8]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 8

I found the text "8" starting at index 0 and ending at index 1.

Enter your regex: [0-4[6-8]]

Enter input string to search: 9

No match found.

Enter your regex: [0-9&&[345]]

Enter input string to search: 3

I found the text "3" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 4

I found the text "4" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 5

I found the text "5" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[345]]

Enter input string to search: 2

No match found.

Enter your regex: [0-9&&[345]]

Enter input string to search: 6

No match found.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 3

No match found.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 4

I found the text "4" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 5

I found the text "5" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [2-8&&[4-6]]

Enter input string to search: 7

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 2

I found the text "2" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 3

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 4

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 5

No match found.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 6

I found the text "6" starting at index 0 and ending at index 1.

Enter your regex: [0-9&&[^345]]

Enter input string to search: 9

I found the text "9" starting at index 0 and ending at index 1.

### Next Lesson PHP Tutorial