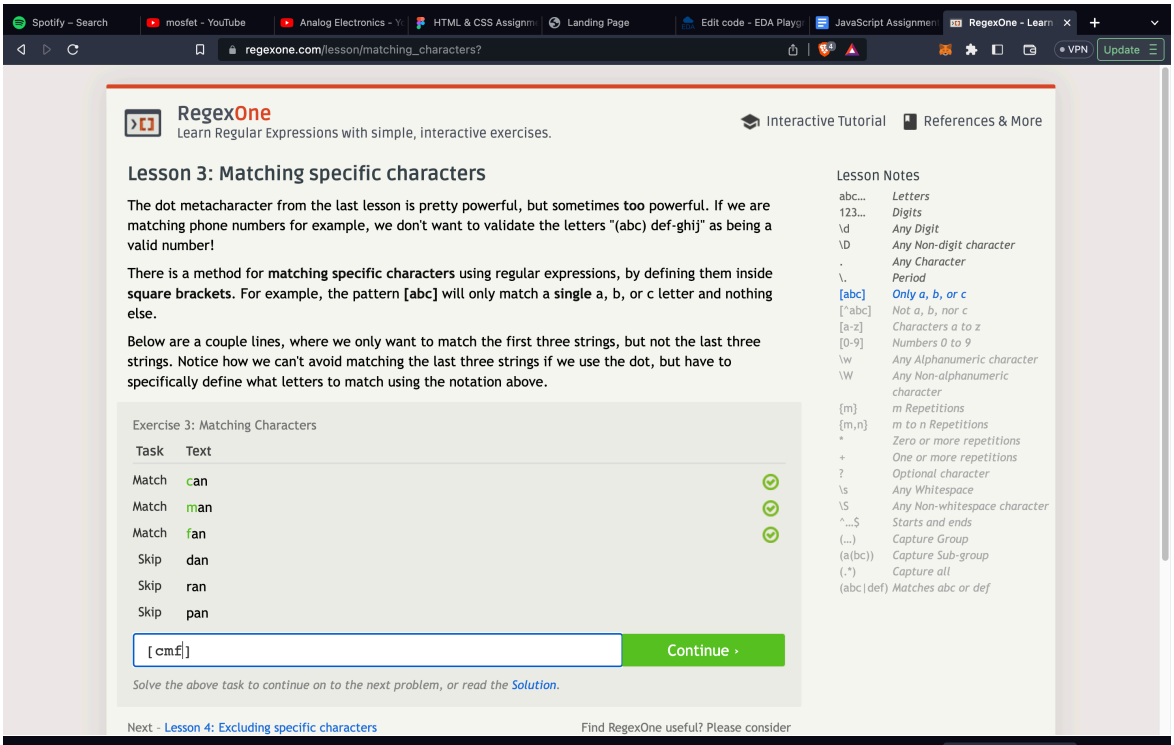


RegexOne
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

Lesson 3: Matching specific characters

The dot metacharacter from the last lesson is pretty powerful, but sometimes **too** powerful. If we are matching phone numbers for example, we don't want to validate the letters "(abc) def-ghij" as being a valid number!

There is a method for **matching specific characters** using regular expressions, by defining them inside **square brackets**. For example, the pattern **[abc]** will only match a **single** a, b, or c letter and nothing else.

Below are a couple lines, where we only want to match the first three strings, but not the last three strings. Notice how we can't avoid matching the last three strings if we use the dot, but have to specifically define what letters to match using the notation above.

Exercise 3: Matching Characters

Task	Text	
Match	can	✓
Match	man	✓
Match	fan	✓
Skip	dan	
Skip	ran	
Skip	pan	

[cmf] Continue ›

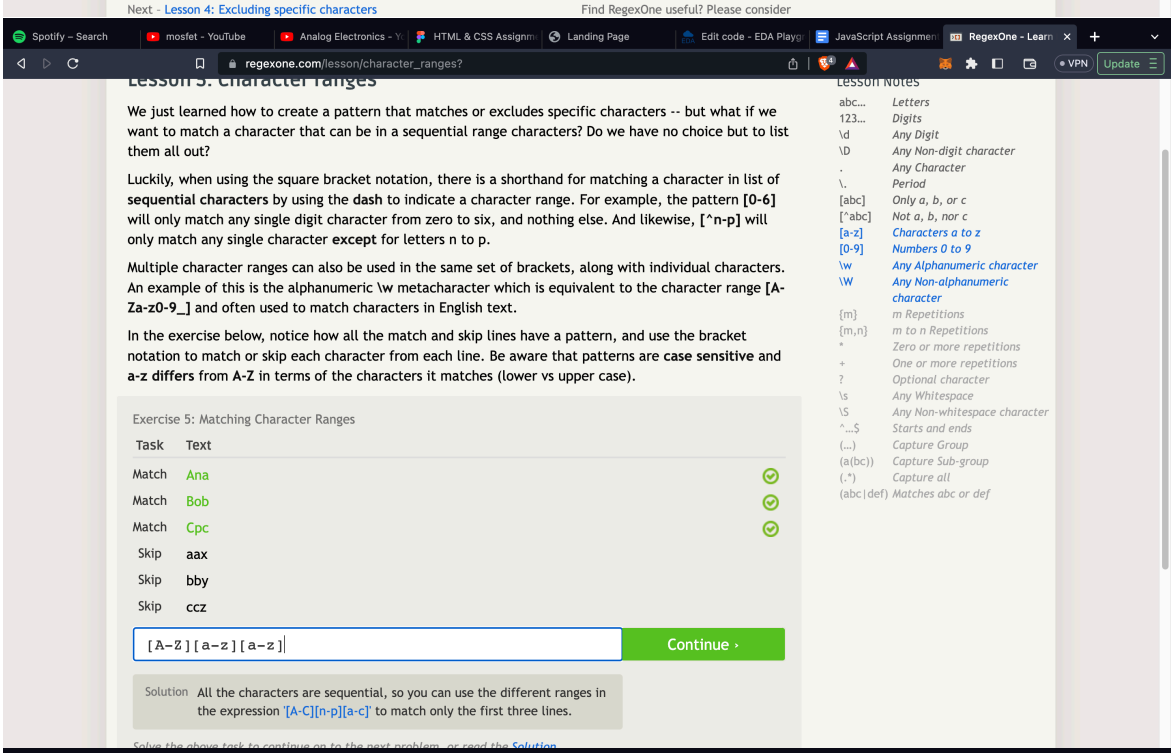
Solve the above task to continue on to the next problem, or read the Solution.

Next - Lesson 4: Excluding specific characters

Find RegexOne useful? Please consider

Lesson Notes

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^...$	Starts and ends
(...)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def



Lesson 5: Character ranges

We just learned how to create a pattern that matches or excludes specific characters -- but what if we want to match a character that can be in a sequential range characters? Do we have no choice but to list them all out?

Luckily, when using the square bracket notation, there is a shorthand for matching a character in list of **sequential characters** by using the **dash** to indicate a character range. For example, the pattern **[0-6]** will only match any single digit character from zero to six, and nothing else. And likewise, **[^n-p]** will only match any single character **except** for letters n to p.

Multiple character ranges can also be used in the same set of brackets, along with individual characters. An example of this is the alphanumeric \w metacharacter which is equivalent to the character range **[A-Za-z0-9_]** and often used to match characters in English text.

In the exercise below, notice how all the match and skip lines have a pattern, and use the bracket notation to match or skip each character from each line. Be aware that patterns are **case sensitive** and **a-z differs** from **A-Z** in terms of the characters it matches (lower vs upper case).

Exercise 5: Matching Character Ranges

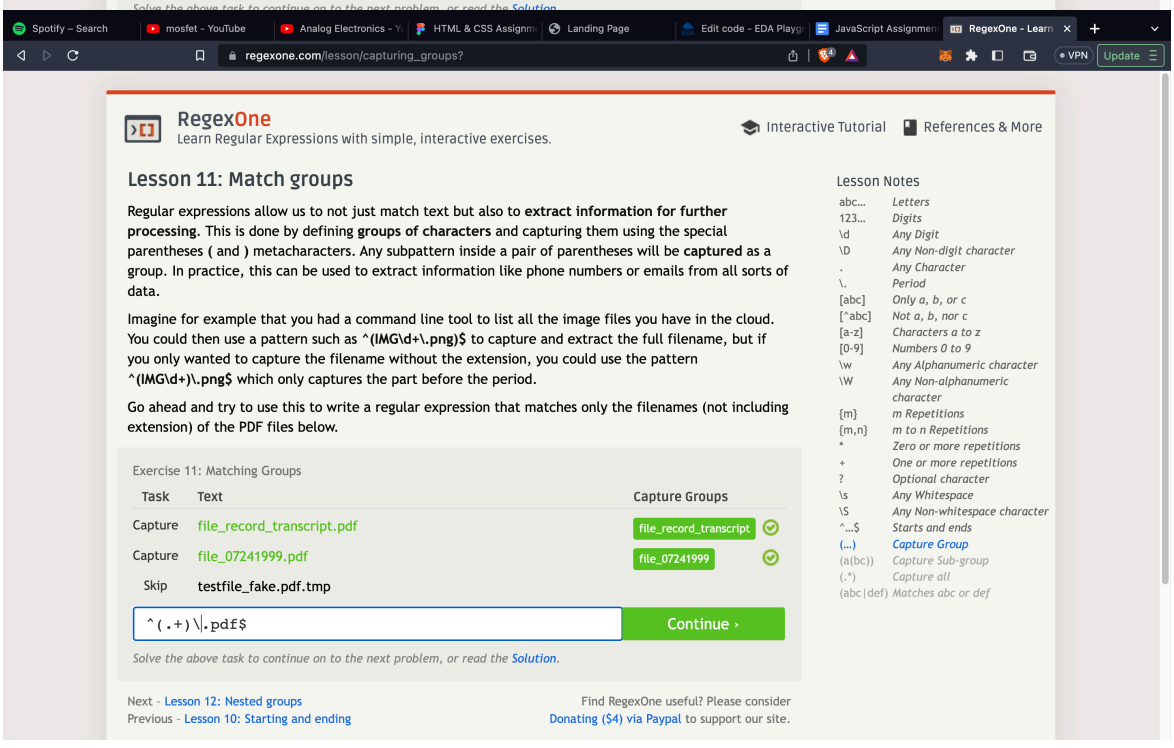
Task	Text	
Match	Ana	✓
Match	Bob	✓
Match	Cpc	✓
Skip	aax	
Skip	bby	
Skip	ccz	

[A-Z][a-z][a-z] Continue ›

Solution All the characters are sequential, so you can use the different ranges in the expression '[A-C][n-p][a-c]' to match only the first three lines.

Lesson Notes

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^...$	Starts and ends
(...)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def



RegexOne
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

Lesson 11: Match groups

Regular expressions allow us to not just match text but also to **extract information for further processing**. This is done by defining **groups of characters** and capturing them using the special parentheses **(** and **)** metacharacters. Any subpattern inside a pair of parentheses will be **captured** as a group. In practice, this can be used to extract information like phone numbers or emails from all sorts of data.

Imagine for example that you had a command line tool to list all the image files you have in the cloud. You could then use a pattern such as **^(IMG\d+\.png)$** to capture and extract the full filename, but if you only wanted to capture the filename without the extension, you could use the pattern **^(IMG\d+)\.png$** which only captures the part before the period.

Go ahead and try to use this to write a regular expression that matches only the filenames (not including extension) of the PDF files below.

Exercise 11: Matching Groups

Task	Text	Capture Groups	
Capture	file_record_transcript.pdf	file_record_transcript	✓
Capture	file_07241999.pdf	file_07241999	✓
Skip	testfile_fake.pdf.tmp		

^(.+)\.pdf$ Continue ›

Solve the above task to continue on to the next problem, or read the Solution.

Next - Lesson 12: Nested groups
Previous - Lesson 10: Starting and ending

Find RegexOne useful? Please consider
Donating ($4) via Paypal to support our site.

Lesson Notes

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^...$	Starts and ends
(...)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def

Spotify – Search | mosfet - YouTube | Analog Electronics - Y | HTML & CSS Assignm | Landing Page | Edit code - EDA Playg | JavaScript Assignmen | RegexOne - Learn

regexone.com/lesson/nested_groups?

RegexOne
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

Lesson 12: Nested groups

When you are working with complex data, you can easily find yourself having to extract multiple layers of information, which can result in nested groups. Generally, the results of the captured groups are in the order in which they are defined (in order by open parenthesis).

Take the example from the previous lesson, of capturing the filenames of all the image files you have in a list. If each of these image files had a sequential picture number in the filename, you could extract both the filename and the picture number using the same pattern by writing an expression like **^(IMG(\d+))\.png$** (using a nested parenthesis to capture the digits).

The nested groups are read from left to right in the pattern, with the first capture group being the contents of the first parentheses group, etc.

For the following strings, write an expression that matches **and captures** both the full date, as well as the year of the date.

Exercise 12: Matching Nested Groups

Task	Text	Capture Groups
Capture	Jan 1987	Jan 1987 | 1987
Capture	May 1969	May 1969 | 1969
Capture	Aug 2011	Aug 2011 | 2011

`(... (\d+))` Continue ›

Solve the above task to continue on to the next problem, or read the Solution.

Next - Lesson 13: More group work
Previous - Lesson 11: Match groups

Find RegexOne useful? Please consider Donating ($4) via Paypal to support our site.

Lesson Notes

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^...$	Starts and ends
(...)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def

Spotify – Search | mosfet - YouTube | Analog Electronics - Y | HTML & CSS Assignm | Landing Page | Edit code - EDA Playg | JavaScript Assignmen | RegexOne - Learn

regexone.com/lesson/conditionals?

As we mentioned before, it's always good to be precise, and that applies to coding, talking, and even regular expressions. For example, you wouldn't write a grocery list for someone to **Buy more .*** because you would have no idea what you could get back. Instead you would write **Buy more milk** or **Buy more bread**, and in regular expressions, we can actually define these conditionals explicitly.

Specifically when using groups, you can use the **| (logical OR, aka. the pipe)** to denote **different possible sets of characters**. In the above example, I can write the pattern "Buy more (milk|bread|juice)" to match only the strings Buy more milk, Buy more bread, or Buy more juice.

Like normal groups, you can use any sequence of characters or metacharacters in a condition, for example, ([cb]ats*|[dh]ogs?) would match either cats or bats, or, dogs or hogs. Writing patterns with many conditions can be hard to read, so you should consider making them separate patterns if they get too complex.

Go ahead and try writing a conditional pattern that matches only the lines with small fuzzy creatures below.

Exercise 14: Matching Conditional Text

Task	Text
Match	I love cats
Match	I love dogs
Skip	I love logs
Skip	I love cogs

`I love (cats|dogs)` Continue ›

Solve the above task to continue on to the next problem, or read the Solution.

Next - Lesson 15: Other special characters
Previous - Lesson 13: More group work

Find RegexOne useful? Please consider Donating ($4) via Paypal to support our site.

Lesson Notes

abc...	Letters
123...	Digits
\d	Any Digit
\D	Any Non-digit character
.	Any Character
\.	Period
[abc]	Only a, b, or c
[^abc]	Not a, b, nor c
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character
{m}	m Repetitions
{m,n}	m to n Repetitions
*	Zero or more repetitions
+	One or more repetitions
?	Optional character
\s	Any Whitespace
\S	Any Non-whitespace character
^...$	Starts and ends
(...)	Capture Group
(a(bc))	Capture Sub-group
(.*)	Capture all
(abc|def)	Matches abc or def

Spotify – Search | mosfet - YouTube | Analog Electronics - Y | HTML & CSS Assignm | Landing Page | Edit code - EDA Playg | JavaScript Assignm | RegexOne - Learn

regexone.com/lesson/end?

RegexOne
Learn Regular Expressions with simple, interactive exercises.

Interactive Tutorial | References & More

Lesson X: Infinity and beyond!

You've finished the tutorial!

Congrats on finishing the lessons! We hope that they've given you a bit more experience with regular expressions and how to apply them in everyday use.

There are still topics within regular expressions that we have not yet explored, - things like greedy vs. non-greedy expressions, posix notation and more. We will try to elaborate more on these in future lessons.