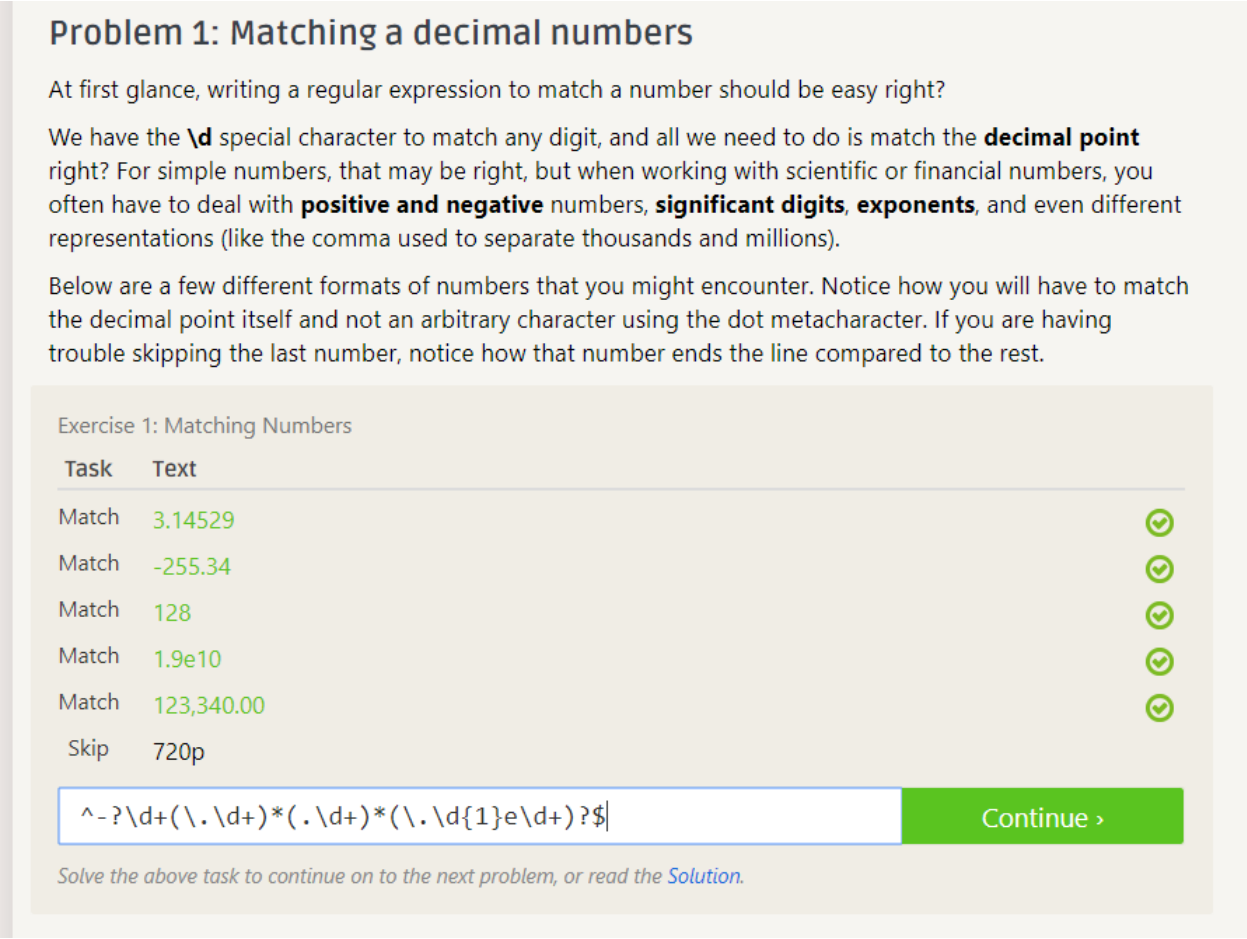
NICK GORECKI

Problem 1



Problem 1: Matching a decimal numbers

At first glance, writing a regular expression to match a number should be easy right?

We have the **\d** special character to match any digit, and all we need to do is match the **decimal point** right? For simple numbers, that may be right, but when working with scientific or financial numbers, you often have to deal with **positive and negative** numbers, **significant digits**, **exponents**, and even different representations (like the comma used to separate thousands and millions).

Below are a few different formats of numbers that you might encounter. Notice how you will have to match the decimal point itself and not an arbitrary character using the dot metacharacter. If you are having trouble skipping the last number, notice how that number ends the line compared to the rest.

Exercise 1: Matching Numbers

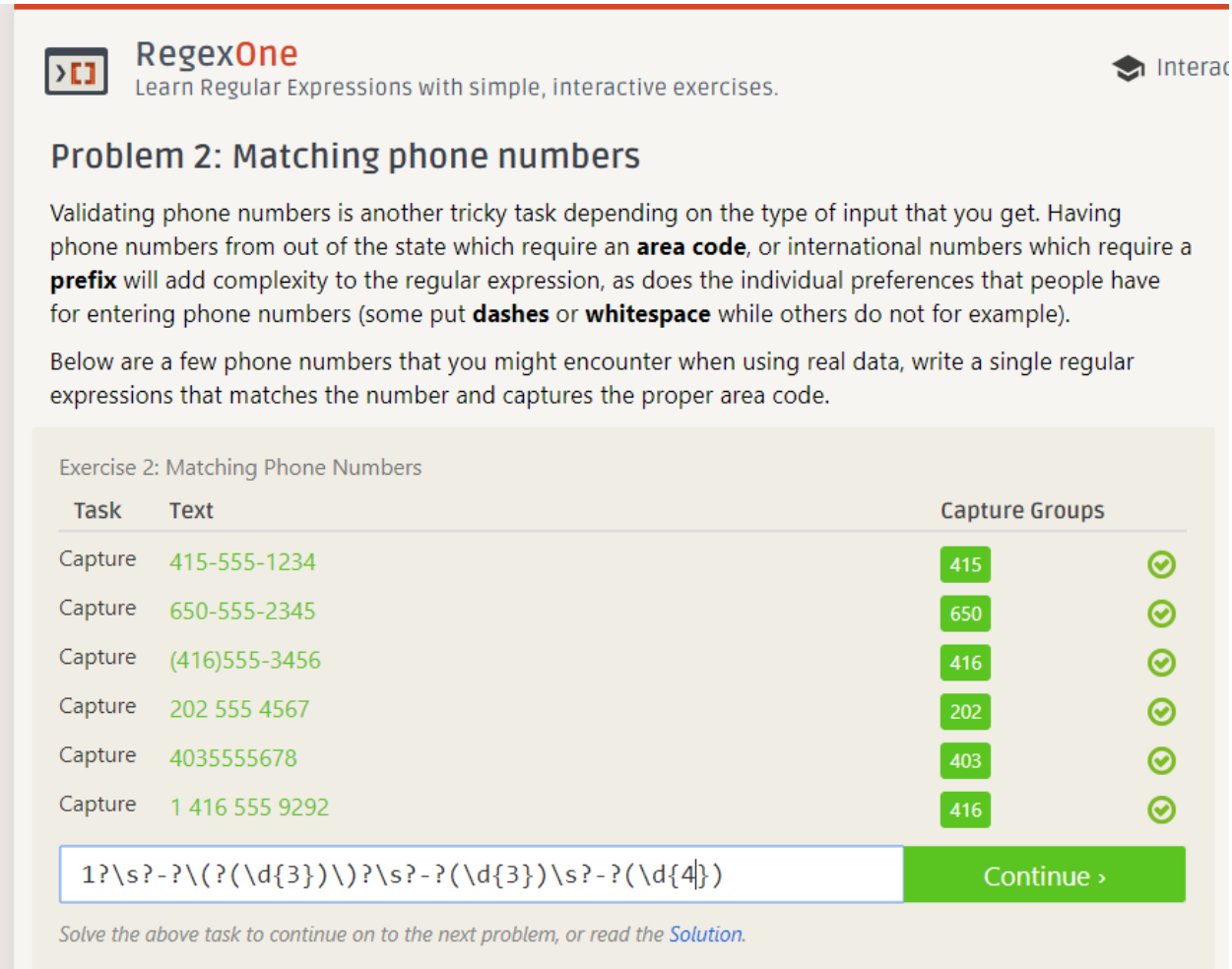
Task	Text
Match	3.14529
Match	-255.34
Match	128
Match	1.9e10
Match	123,340.00
Skip	720p

```
^-?\d+(\.\d+)*(.\d+)*(\.\d{1}e\d+)?$
```

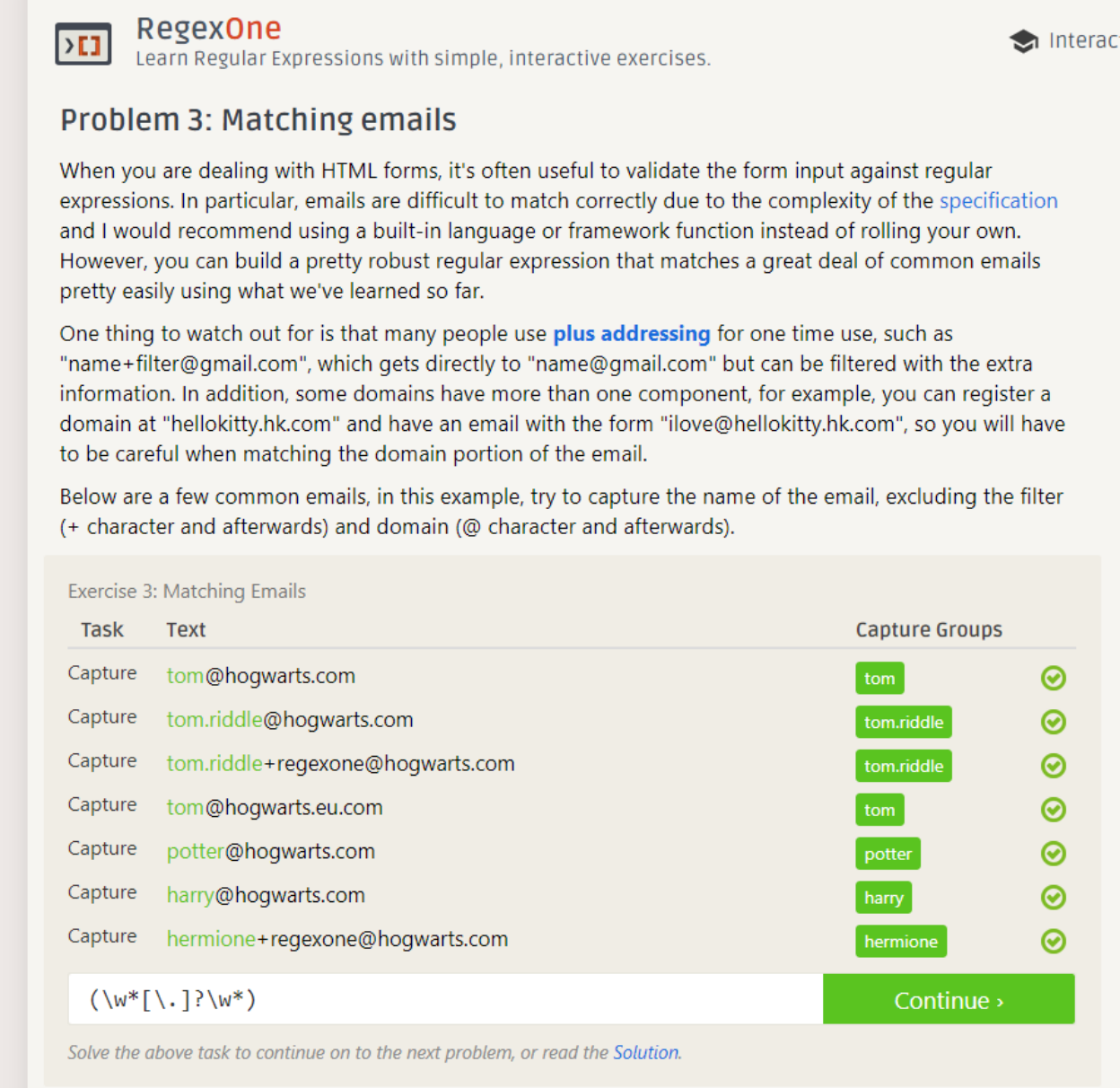
Continue ›

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

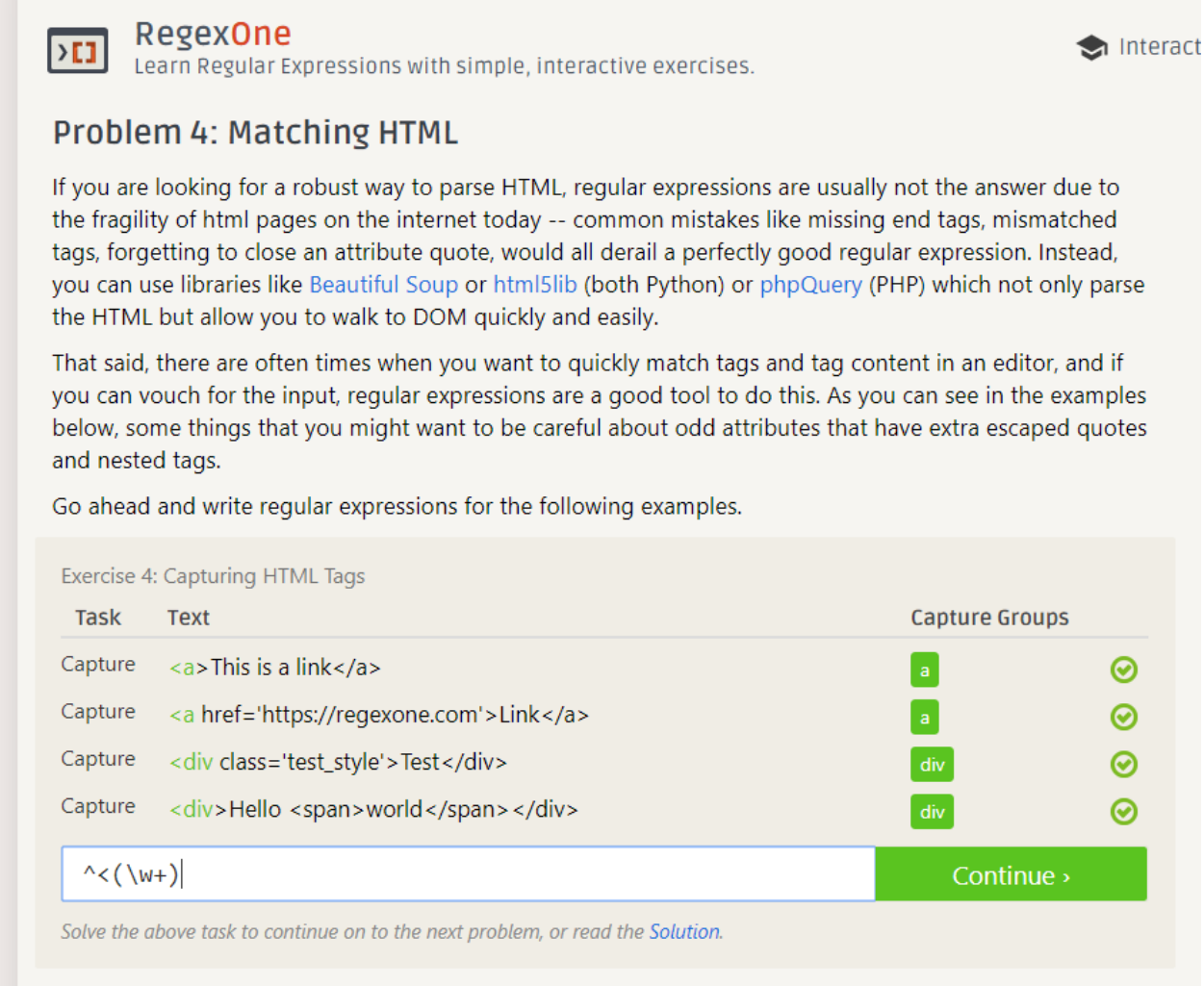
Problem 2


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interac
Problem 2: Matching phone numbers
Validating phone numbers is another tricky task depending on the type of input that you get. Having phone numbers from out of the state which require an area code, or international numbers which require a prefix will add complexity to the regular expression, as does the individual preferences that people have for entering phone numbers (some put dashes or whitespace while others do not for example).
Below are a few phone numbers that you might encounter when using real data, write a single regular expressions that matches the number and captures the proper area code.
Exercise 2: Matching Phone Numbers
Task
Text
Capture Groups
Capture 415-555-1234 415
Capture 650-555-2345 650
Capture (416)555-3456 416
Capture 202 555 4567 202
Capture 4035555678 403
Capture 1 416 555 9292 416
1?\s?-?\(?(\d{3})\)?\s?-?(\d{3})\s?-?(\d{4})
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

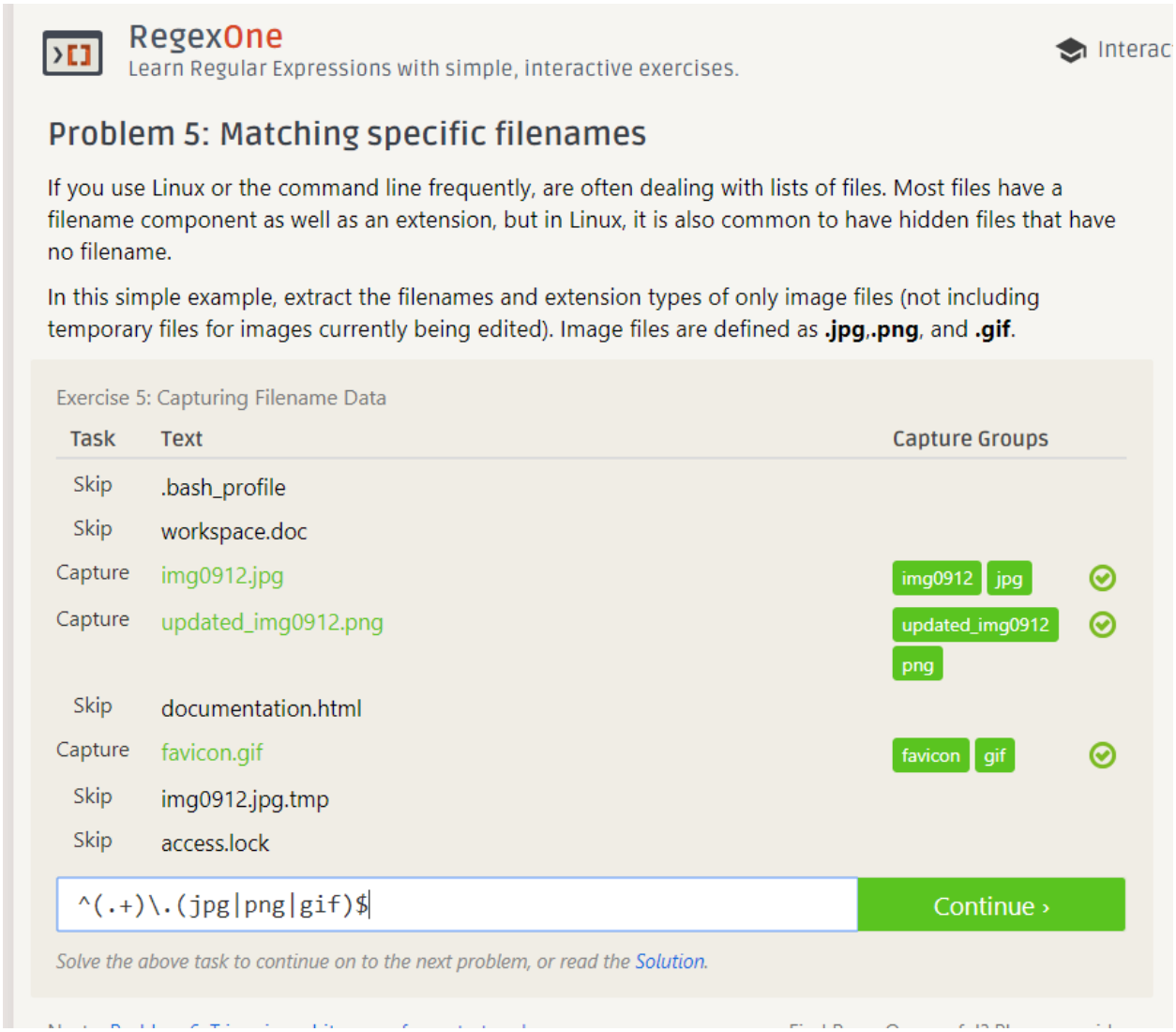
Problem 3


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interac
Problem 3: Matching emails
When you are dealing with HTML forms, it's often useful to validate the form input against regular expressions. In particular, emails are difficult to match correctly due to the complexity of the specification and I would recommend using a built-in language or framework function instead of rolling your own. However, you can build a pretty robust regular expression that matches a great deal of common emails pretty easily using what we've learned so far.
One thing to watch out for is that many people use plus addressing for one time use, such as "name+filter@gmail.com", which gets directly to "name@gmail.com" but can be filtered with the extra information. In addition, some domains have more than one component, for example, you can register a domain at "hellokitty.hk.com" and have an email with the form "ilove@hellokitty.hk.com", so you will have to be careful when matching the domain portion of the email.
Below are a few common emails, in this example, try to capture the name of the email, excluding the filter (+ character and afterwards) and domain (@ character and afterwards).
Exercise 3: Matching Emails
Task
Text
Capture Groups
Capture
tom@hogwarts.com
tom
Capture
tom.riddle@hogwarts.com
tom.riddle
Capture
tom.riddle+regexone@hogwarts.com
tom.riddle
Capture
tom@hogwarts.eu.com
tom
Capture
potter@hogwarts.com
potter
Capture
harry@hogwarts.com
harry
Capture
hermione+regexone@hogwarts.com
hermione
(\w*[\.]?\w*)
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

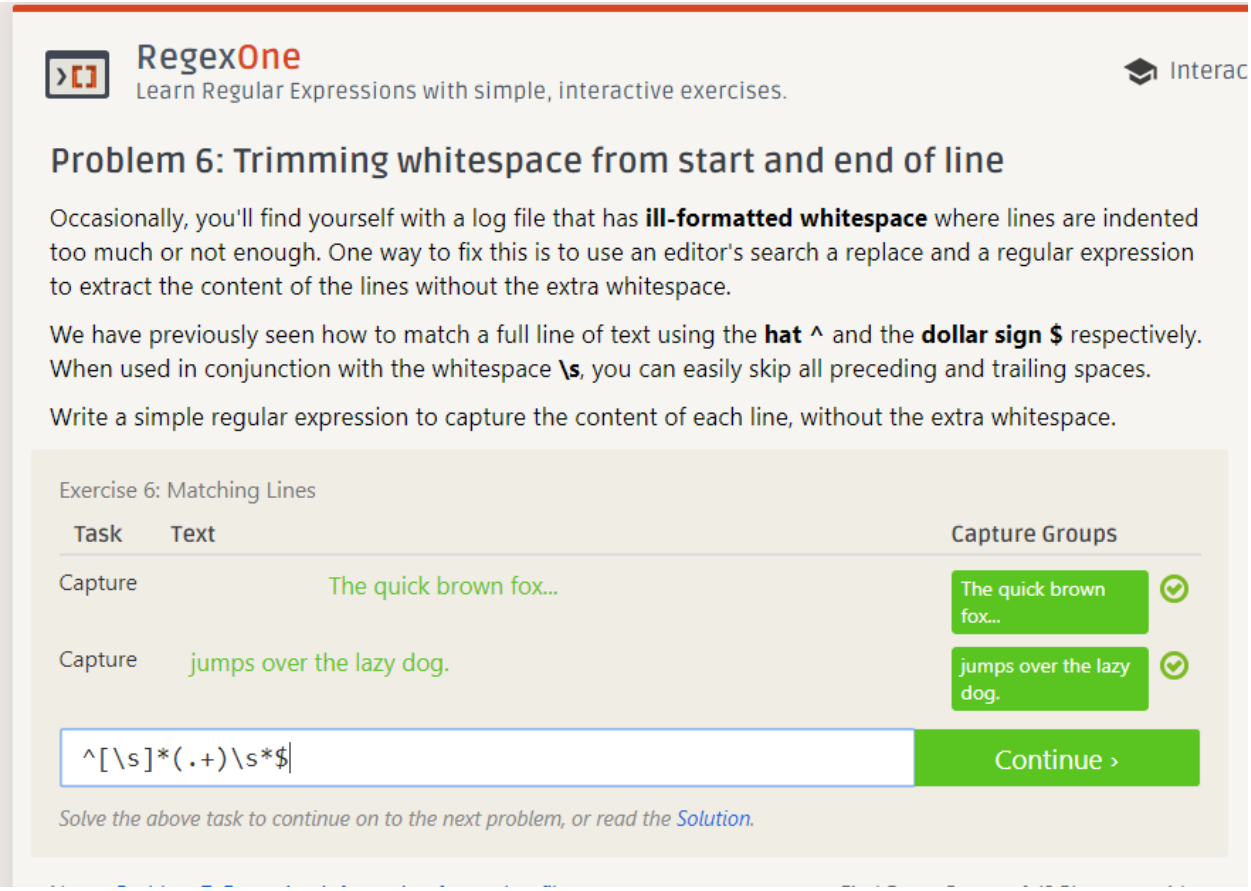
Problem 4


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interact
Problem 4: Matching HTML
If you are looking for a robust way to parse HTML, regular expressions are usually not the answer due to the fragility of html pages on the internet today -- common mistakes like missing end tags, mismatched tags, forgetting to close an attribute quote, would all derail a perfectly good regular expression. Instead, you can use libraries like Beautiful Soup or html5lib (both Python) or phpQuery (PHP) which not only parse the HTML but allow you to walk to DOM quickly and easily.
That said, there are often times when you want to quickly match tags and tag content in an editor, and if you can vouch for the input, regular expressions are a good tool to do this. As you can see in the examples below, some things that you might want to be careful about odd attributes that have extra escaped quotes and nested tags.
Go ahead and write regular expressions for the following examples.
Exercise 4: Capturing HTML Tags
Task
Text
Capture Groups
Capture
<a>This is a link
a
Capture
Link
a
Capture
Test
div
Capture
Hello world
div
^<(\w+)
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

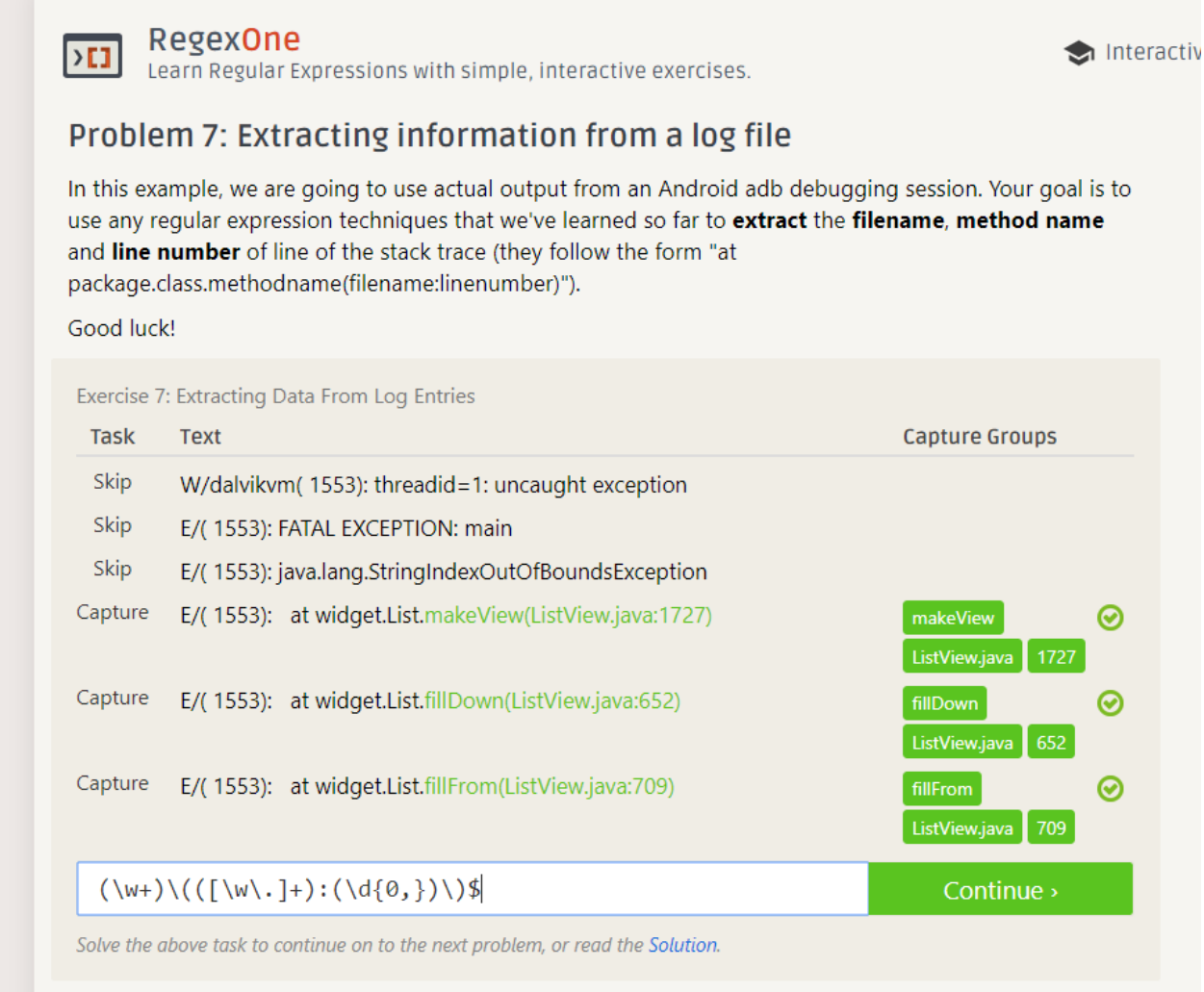
Problem 5


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interac
Problem 5: Matching specific filenames
If you use Linux or the command line frequently, are often dealing with lists of files. Most files have a filename component as well as an extension, but in Linux, it is also common to have hidden files that have no filename.
In this simple example, extract the filenames and extension types of only image files (not including temporary files for images currently being edited). Image files are defined as .jpg,.png, and .gif.
Exercise 5: Capturing Filename Data
Task
Text
Capture Groups
Skip
.bash_profile
Skip
workspace.doc
Capture
img0912.jpg
img0912
jpg
Capture
updated_img0912.png
updated_img0912
png
Skip
documentation.html
Capture
favicon.gif
favicon
gif
Skip
img0912.jpg.tmp
Skip
access.lock
^(.+)\.(jpg|png|gif)$
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

Problem 6


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interac
Problem 6: Trimming whitespace from start and end of line
Occasionally, you'll find yourself with a log file that has ill-formatted whitespace where lines are indented too much or not enough. One way to fix this is to use an editor's search a replace and a regular expression to extract the content of the lines without the extra whitespace.
We have previously seen how to match a full line of text using the hat ^ and the dollar sign $ respectively. When used in conjunction with the whitespace \s, you can easily skip all preceding and trailing spaces.
Write a simple regular expression to capture the content of each line, without the extra whitespace.
Exercise 6: Matching Lines
Task
Text
Capture Groups
Capture
The quick brown fox...
The quick brown fox...
Capture
jumps over the lazy dog.
jumps over the lazy dog.
^[\s]*(.+)\s*$
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

Problem 7


RegexOne
Learn Regular Expressions with simple, interactive exercises.
Interactiv
Problem 7: Extracting information from a log file
In this example, we are going to use actual output from an Android adb debugging session. Your goal is to use any regular expression techniques that we've learned so far to extract the filename, method name and line number of line of the stack trace (they follow the form "at package.class.methodname(filename:linenumber)").
Good luck!
Exercise 7: Extracting Data From Log Entries
Task
Text
Capture Groups
Skip
W/dalvikvm(1553): threadid=1: uncaught exception
Skip
E/(1553): FATAL EXCEPTION: main
Skip
E/(1553): java.lang.StringIndexOutOfBoundsException
Capture
E/(1553): at widget.List.makeView(ListView.java:1727)
makeView
ListView.java
1727
Capture
E/(1553): at widget.List.fillDown(ListView.java:652)
fillDown
ListView.java
652
Capture
E/(1553): at widget.List.fillFrom(ListView.java:709)
fillFrom
ListView.java
709
(\w+)\(([\w\.]+):(\d{0,})\)$
Continue ›
Solve the above task to continue on to the next problem, or read the Solution.

Problem 8

Learn Regular Expressions with simple, interactive exercises.

Interactiv

Problem 8: Parsing and extracting data from a URL

When working with files and resources over a network, you will often come across URIs and URLs which can be parsed and worked with directly. Most standard libraries will have classes to parse and construct these kind of identifiers, but if you need to match them in logs or a larger corpus of text, you can use regular expressions to pull out information from their structured format quite easily.

URIs, or Uniform Resource Identifiers, are a representation of a resource that is generally composed of a scheme, host, port (optional), and resource path, respectively highlighted below.

```
http://regexone.com:80/page
```

The scheme describes the protocol to communicate with, the host and port describe the source of the resource, and the full path describes the location at the source for the resource.

In the exercise below, try to extract the protocol, host and port of the all the resources listed.

Exercise 8: Extracting Data From URLs

Task	Text	Capture Groups
Capture	ftp://file_server.com:21/top_secret/life_changing_plans.pdf	ftp, file_server.com, 21
Capture	https://regexone.com/lesson/introduction#section	https, regexone.com
Capture	file://localhost:4040/zip_file	file, localhost, 4040
Capture	https://s3cur3-server.com:9999/	https, s3cur3-server.com, 9999
Capture	market://search/angry%20birds	market, search

```
^(\w+)://([\w-\.]*)/?(:(\d*))?
```

Continue ›

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