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 python code using regular expression(s) that matches all of the numbers.

212.445.6765

202-239-2454

(404) 864-5689

650 678 2134

7733124571

Hint – putting the data in a list or using the input function and looping through may save some time for these problems.

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.

Bill.Gates@microsoft.com

president@whitehouse.gov

Mark@facebook.com

postoffice22030@postal.gov

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.

Write a simple regular expression to capture the content of each line, without the extra whitespace.

```
“                              The quick brown fox...”
“jumps over the lazy dog.                           “
```

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.

ftp://file_server.com:21/top_secret/life_changing_plans.pdf

https://regone.com/lesson/introduction#section

file://localhost:4040/zip_file

https://s3cur3-server.com:9999/

Using regular expressions determine write code to print an alert if it detects either a file URL a URL that contains port 9999. Use the URLs above as test data