Overview

Regular expressions (regex or regexp for short) are special sequences of characters that define patterns
to search for in text. They're often used in find-and-replace operations, or to add up the number of words
or phrases matching a particular pattern.

Regular expressions are useful in a variety of applications, and can be used in different programs and
programming languages. We will start by learning the general components of regular expressions, using a
simple online tool, RegExr. Then at the end of the workshop, we'll learn how to use regular expressions
in a text editor, Sublime. We'll also demonstrate how to use them in Python and R, for those students
already familiar with one of those languages.

To get started:

1. Go to this site: http://regexr.com (http://regexr.com).
2. Copy and paste the New York Times leads from the file `nyt_leads.txt` into the **Text** window on the website.
3. Delete what you see in the **Expression** field. This is where we'll insert our own regular expressions
   to find sequences in the headlines below.

```
New York Times                                          October 19, 2016
Retaking Mosul From ISIS May Pale to What Comes Next
By TIM ARANGO and RICK GLADSTONE  11:52 ET
If the recaptures of Ramadi, Tikrit and Falluja are a guide,
Iraqi officials will confront devastation and unexploded bombs
once Mosul is reclaimed.

New York Times                                          October 18, 2016
Short-Term Cease-Fire in Yemen Appears Likely
By BEN HUBBARD  10:05 ET
The rebels known as the Houthis said they would abide by the
cease-fire if the Saudi military coalition halted attacks and
lifted a blockade.
```

1. Special Characters

Strings are composed of characters, and we are writing patterns to match specific sequences of characters.
Various characters have special meaning in regular expressions. When we use these characters in an expression,
we aren't matching the identical character, we're using the character as a placeholder for some other character(s)
or part(s) of a string.

If you want to match a character that happens to be a special character, you have to escape it with a backslash
\. Try typing the following special characters into the **Expression** field on the regexr.com site. What happens
when you type New York Times vs. ^New York Times? How about ., \., or \.$?

```
.          any single character
^          start of string
$          end of string
\n         new line
\r         carriage return
\t         tab
```

2. Quantifiers

Some special characters refer to optional characters, to a specific number of characters, or to an open-ended
number of characters matching the preceding pattern. Try looking for the letter 'o' followed by a number of 'f's:
what happens if you type of, of*, of+, of{1}, of{1,2}?

```
*          0 or more of the preceding character/expression
+          1 or more of the preceding character/expression
?          0 or 1 of the preceding character/expression
{n}        n copies of the preceding character/expression
{n,m}      n to m copies of the preceding character/expression
```

2. Sets

Regular expressions also allow you to define sets of characters. Within a set of square brackets, you may list
characters individually, e.g. [aeiou], or in a range, e.g. [A-Z] (note that all regular expressions are case
sensitive).

You can also create a complement set by excluding certain characters, using ^ as the first character
in the set. The set [^A-Za-z] will match any character except a letter. All other special characters loose
their special meaning inside a set, so the set [.?] will look for a literal period or question mark.

The set will match only one character contained within that set, so to find sequences of multiple characters from
the same set, use a quantifier like + or a specific number or number range {n,m}.

```
[0-9]          any numeric character
[a-z]          any lowercase alphabetic character
[A-Z]          any uppercase alphabetic character
[aeiou]        any vowel (i.e. any character within the brackets)
[0-a-z]      to combine sets, list them one after another
[^...]         exclude specific characters
```

3. Special sequences

Several special characters denote special sequences. These begin with a \ followed by a letter. Note that the uppercase version is usually the complement of the lowercase version.

```
\d          Any digit
\D          Any non-digit character
\w          Any alphanumeric character [0-9a-zA-Z_]
\W          Any non-alphanumeric character
\s          Any whitespace (space, tab, new line)
\S          Any non-whitespace character
\b          Matches the beginning or end of a word (does not consume a character)
\B          Matches only when the position is not the beginning or end of a word
(does not consume a character)
```

4. Groups and Logical OR

Parentheses are used to designate groups of characters, to aid in logical conditions, and to be able to retrieve the
contents of certain groups separately.

The pipe character | serves as a logical OR operator, to match the expression before or after the pipe. Group parentheses
can be used to indicate which elements of the expression are being operated on by the |.

```
|             Logical OR opeator
(...)         Matches whatever regular expression is inside the parentheses, and
notes the start and end of a group
(this|that)   Matches the expression "this" or the expression "that"
```