Transformation and Cleaning with Regular Expressions & ETL Mini Project

Data Boot Camp

Lesson 13.2

Class Objectives

By the end of today's lesson, you will be able to:

Use wildcards, sets, and escape characters in regular expressions to extract information.

Use special characters in regular expressions to extract information.

Create and use capture groups in regular expressions to extract information.

Continue working on the ETL mini project.

Sets and Wildcards, and Escaping

Instructor Demonstration

Wildcards, Sets, and Escaping

Wildcards

Wildcards allow us to match different types of characters (like letters, digits, or white space characters). For example, the dot wildcard (.) allows us to match any character.

```
# Find all lines of text that start with any character
and then include 'ought' elsewhere in the line.
p = '.ought'
sample_df[sample_df['text'].str.contains(p)]]
# Will return matches like bought, $ought, and 4ought.
```

Wildcards

The backslash followed by lowercase w, (\w) is a wildcard that matches any letter, digit, or underscore.

```
# Use \w to find any letter, digit, or underscore
followed by ought.
p = '\wought'
sample_df[sample_df['text'].str.contains(p)]]
# Will return matches like bought, thought, fought,
sought, 55ought, and _ought.
```

Sets

We can use brackets to match specific characters, which is called a **set**.
For example, we can use the [bfs] set in [bfs]ought to match 'bought', 'fought', and 'sought'.

```
# Find all lines of text with the strings 'bought',
'fought', and 'sought'.
p = '[bfs]ought'
sample_df[sample_df['text'].str.contains(p)]]
# Will return matches of bought, fought, and sought.
```

Escaping

To match a character that also happens to be a regular expression character, we add a backslash (\) as an **escape character**. For example, to match a period (.) in a sentence, we use a backslash followed by a period (\.).

```
# Find all lines of text with the strings 'bought',
'fought', and 'sought' that end with a period
# note: we need to escape the period with a backslash.
p = '[bfs]ought\.'
sample_df[sample_df['text'].str.contains(p)]]
# Will return matches for 'sought.'
```

Questions?

Activity: Wildcards, Sets, and Escaping

In this activity, you'll use regular expressions to find lines of text that meet specific criteria.

Suggested Time:

15 minutes

Time's Up! Let's Review.

Questions?

Regex Special Characters

Instructor Demonstration

Regex Special Characters

Regex Special Characters: Question Mark (?)

The question mark (?) lets us match either none or one of the preceding characters.

```python
# Find all lines of text that contain hear or heard.
p = 'heard?'
sherlock_df[sherlock_df['text'].str.contains(p)]
# Will return matches for both hear and heard.
```

Regex Special Characters: Asterisk (*)

The asterisk (*) lets us match either none, one, or more than one of the preceding characters.

```python
# Find all lines of text that contain tel, tell, or
telll
p = 'tell*'
sherlock_df[sherlock_df['text'].str.contains(p)]
# Will return matches for 'tel', 'tell', 'telll', and
so on.
```

Regex Special Characters: Caret (^)

The caret (^) lets us match lines that start with the subsequent expression.

```python
# Find all lines of text that start with the string
'Watson'.
p = '^Watson'
sherlock_df[sherlock_df['text'].str.contains(p)]
# Will return matches for lines like 'Watson said
this', but not for 'I told Watson'.
```

Regex Special Characters: Dollar Sign ($)

The dollar sign ($) lets us match lines that end with the preceding expression.

```
# Find all lines of text that end with a period.
p = '\.$'
sherlock_df[sherlock_df['text'].str.contains(p)]
# Will return matches for lines like 'Watson ran.'
but not for 'Watson and I will follow in the second."'
```

Regex Special Characters: Pipe (|)

The pipe (|) lets us put a conditional expression in our regex expression to match the term that either precedes or follows it.

```
# Use | to match lines that end with either a period
or a question mark.
p = '\.$|\?$'
sherlock_df[sherlock_df['text'].str.contains(p)]
# Will return matches for both of the following lines:
 'Watson ran.' and 'What did Holmes say?'
```

Questions?

Activity: Regex Special Characters

In this activity, you'll use special characters to find lines of text that meet specific criteria.

Suggested Time:

15 minutes

Time's Up! Let's Review.

Questions?

Regex Grouping

Additional Regular Expression Techniques
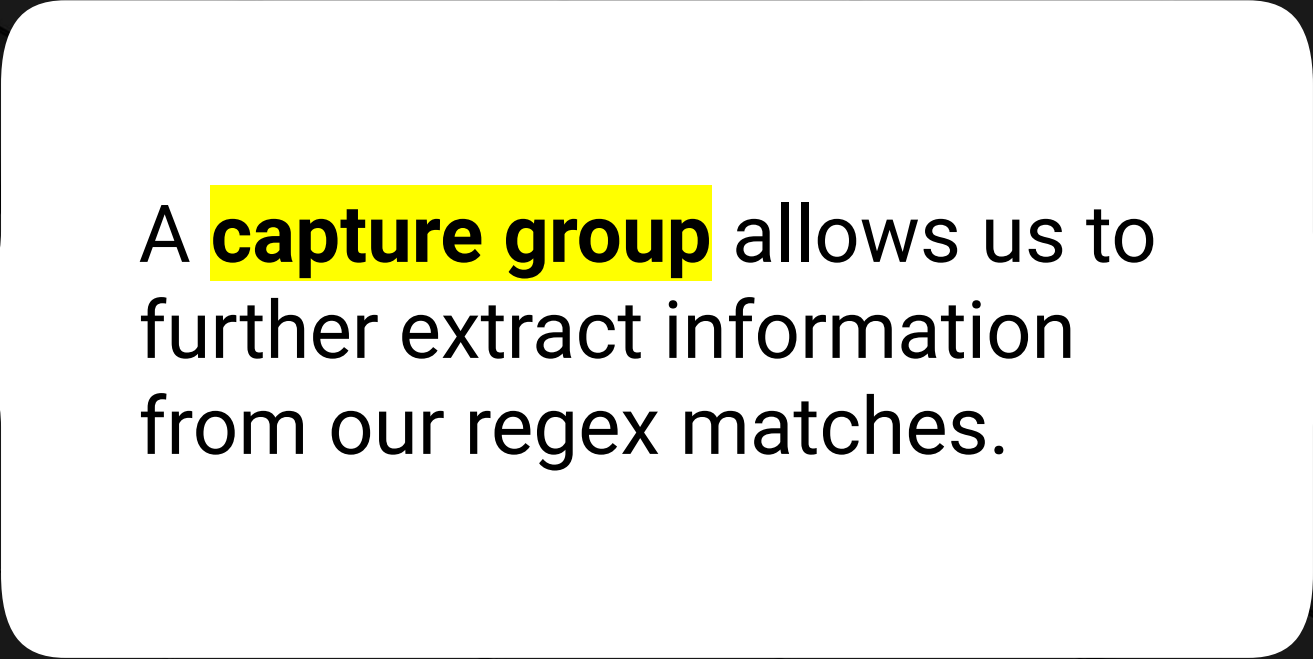
Before getting into grouping, let's introduce a few more regular expression techniques.

`\s`	Matches a white space character.
`{4}`	Matches the preceding regular expression 4 times.
`{4,}`	Matches the preceding regular expression 4 or more times.
`{4, 6}`	Matches the preceding regular expression between 4 and 6 times (inclusive).

Instructor Demonstration

Regex Grouping

A **capture group** allows us to further extract information from our regex matches.

Questions?

Activity: Regex Grouping

In this activity, you'll use capture groups to further refine regex matches.

Time's Up! Let's Review.

Transforming and Cleaning Data with Regexes

Group Programming Activity:

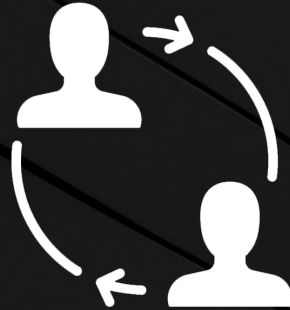# Transforming and Cleaning IoT Data

In this activity, you'll combine your skills in data transformation via Python and Pandas methods with those in regular expressions to transform an Internet of Things (IoT) dataset into a DataFrame.

Suggested Time:

15 Minutes

Questions?

Project Work

Suggested Time:

40 Minutes