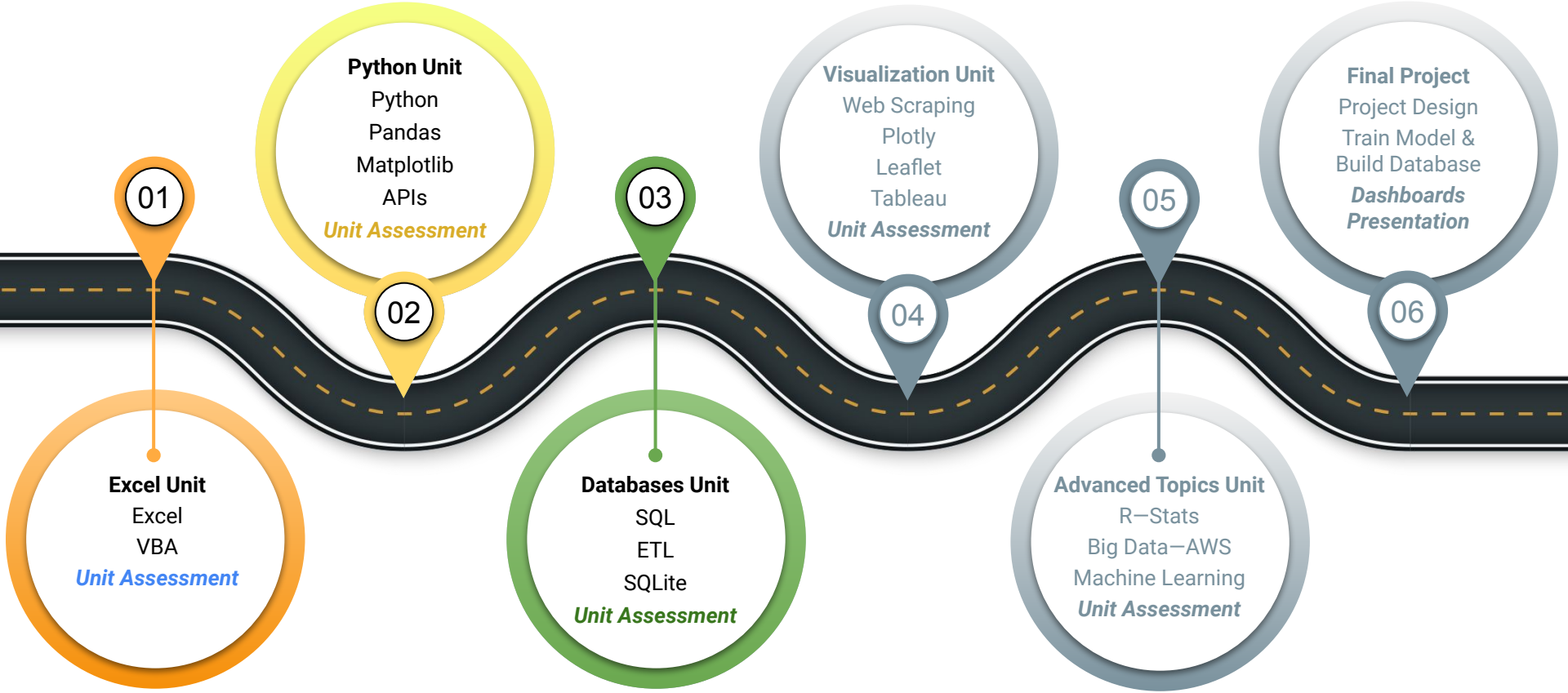
ETL

Data Boot Camp

Lesson 8.2

The Big Picture



This Week: Extract, Transform, and Load (ETL)

By the end of this week, you'll be able to:

01 Use regular expressions to find patterns in string data.

02 Use sets, wildcards, and escaping in regular expressions.

03 Use special characters in regular expressions to find and extract string data.

04 Use regular expression capture groups to retrieve specific information from string data.

05 Transform and clean data by using regular expressions.

This Week’s Challenge

You’ll perform the ETL process on a new dataset. You’ll have the option of using Pandas functions, list comprehensions, and regular expressions. You will then load the data into a PostgreSQL database.

Module 08

Today’s Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

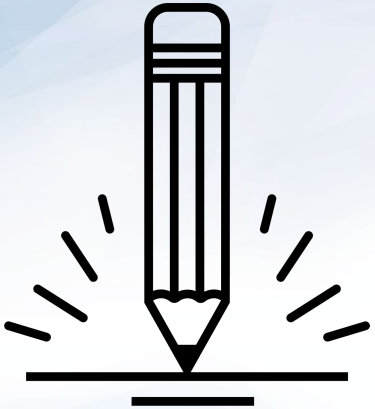
01 Use regex sets, wildcards, and escaping.

02 Use regex special characters.

03 Use regex grouping.

04 Use Pandas functions and regex to extract, transform, and clean data.

Make sure you've downloaded any relevant class files!



Group Activity: Regex Matching with Pandas

In this activity, you'll load a text dataset from *Sherlock Holmes* and then use the Pandas str.contains() function to find text that contains matching patterns.

Suggested Time:
10 minutes

Sets, Wildcards, and Escaping

Wildcards

Wildcards let us match different types of characters (like letters, digits, or white space characters). The dot wildcard (.) matches any character.

```
# Find all lines of text that start with any character
then include 'ought'.
p = '.ought'
# Will return matches like bought, $ought, 4ought
```

Sets

A set lets us match any character that it contains by using brackets.

```
# Simple Function with no parameters
p = '[bfs]ought'
# Will return matches of bought, fought, and sought
```

Escaping

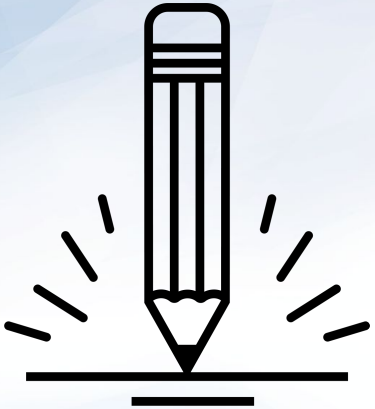
Escaping is a way to match characters that regex itself uses, like the dot (`.`). We do this by preceding the character that we want to escape with the backslash (`\`).

```
# Simple Function with no parameters
p = 'bought\.'
# Will return matches for 'bought.'
```

Instructor Demonstration

Sets, Wildcards, and Escaping

Questions?



Activity:
Sets, Wildcards, and Escaping

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

Suggested Time:
15 minutes

Let’s Review

Regex Special Characters

Questions?

Regex Special Characters: Question Mark (?)

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

```
# Find all lines of text that contain hear or heard.
p = 'heard?'
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.

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

Regex Special Characters: Caret (^)

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

```
# Find all lines of text that start with Watson
p = '^Watson'
str.contains(p)
# Will return matches for line 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 = '\.$'
str.contains(p)
# Will return matches for line like 'Watson ran.',
but not for ‘What did Holmes say?’
```

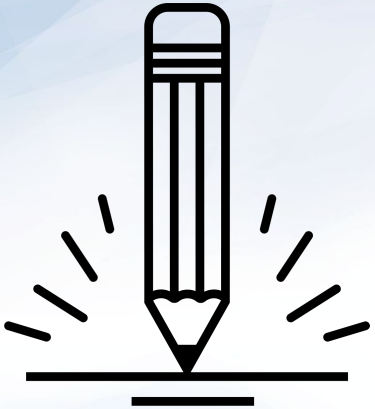
Regex Special Characters: Pipe (|)

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

```
# Find all lines of text that end with a period or a question mark.
p = '\.$|\?$'
str.contains(p)
# Will return matches for both lines 'Watson ran.', and 'What did Holmes say?'
```

Instructor Demonstration

Special Characters



Activity: Special Characters

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

Suggested Time:
15 minutes

Let's Review

Regex Grouping

Additional Regular Expressions

Before discussing grouping, let's learn a few more regular expression techniques.

`\s`	Matches white space.
`{4}`	Matches words that are exactly 4 characters long.
`{4,}`	Matches words that are 4 or more characters long.
`{4, 6}`	Matches words that are 4–6 characters long.

Grouping allows us to further extract information from our regex matches.

Grouping

To find all the words that consist of 4–6 letters and that appear after the word `Holmes`, we search with a regex that matches both. But, we place the word `Holmes` in one group and the word of 4–6 letters in the other. To do so, we place each regex inside parentheses. When dealing with groups, we also use `extractall()`.

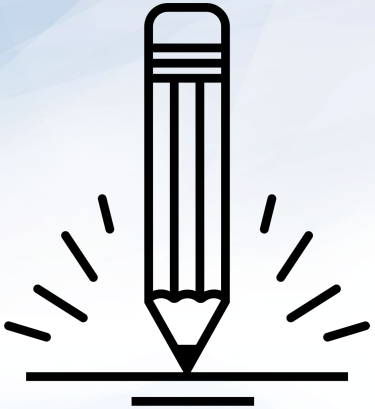
```
# Find all lines of text that contain Holmes followed
by a space and 4 letter word
p = '(Holmes)(\s\w{4,6})'
str.extractall(p)
```

Group Matches

The previous search will return groups like the following:

Group 1	Group 2
`Holmes`	walks
`Holmes`	runs
`Holmes`	sings
`Holmes`	jumping

Questions?



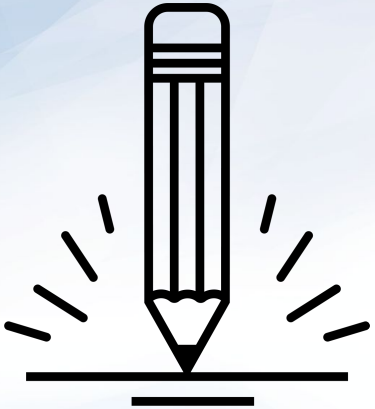
Partner Activity: Groups

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

Suggested Time:
15 minutes

Let's Review

Transforming and Cleaning with Regex



Group Activity: Transforming and Cleaning IoT Data

In this activity, you'll combine your skills in data transformation (using Python and Pandas methods) with those in regex to transform an internet of things (IoT) dataset.

Suggested Time:
20 minutes

Questions?