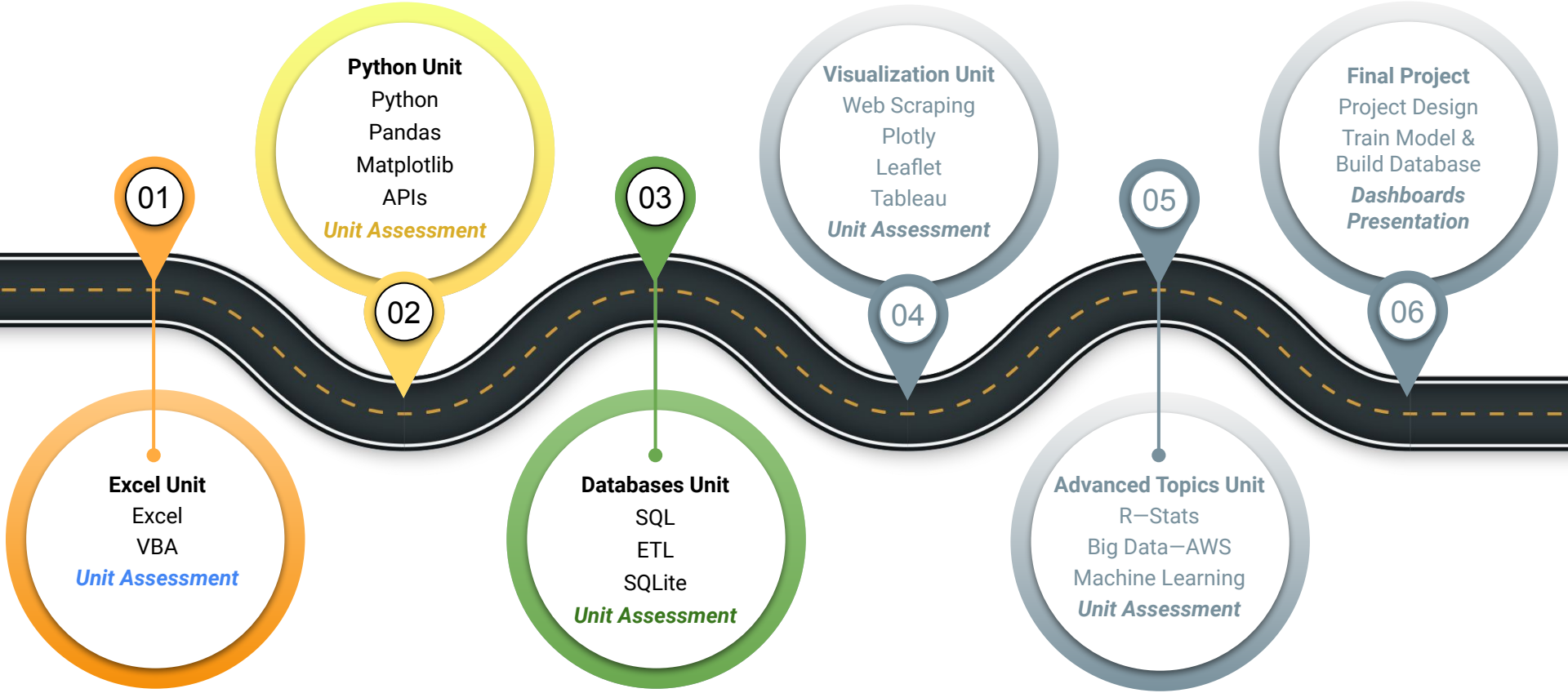
ETL

Data Boot Camp

Lesson 8.2

The Big Picture



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

By the end of this week, you'll know how to:

01 Create an ETL pipeline from raw data to a SQL database.

02 Extract data from disparate sources using Python.

03 Clean and transform data using Pandas.

04 Use regular expressions to parse data and to transform text into numbers.

05 Load data with PostgreSQL.

This Week's Challenge

Using the skills learned throughout the week, perform the ETL process across three different data files while using functions, list comprehension, and regular expressions.

Module 08

Today’s Agenda

Today's Agenda

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

01 Regex Sets, Wildcards, and Escaping

02 Regex Special Characters

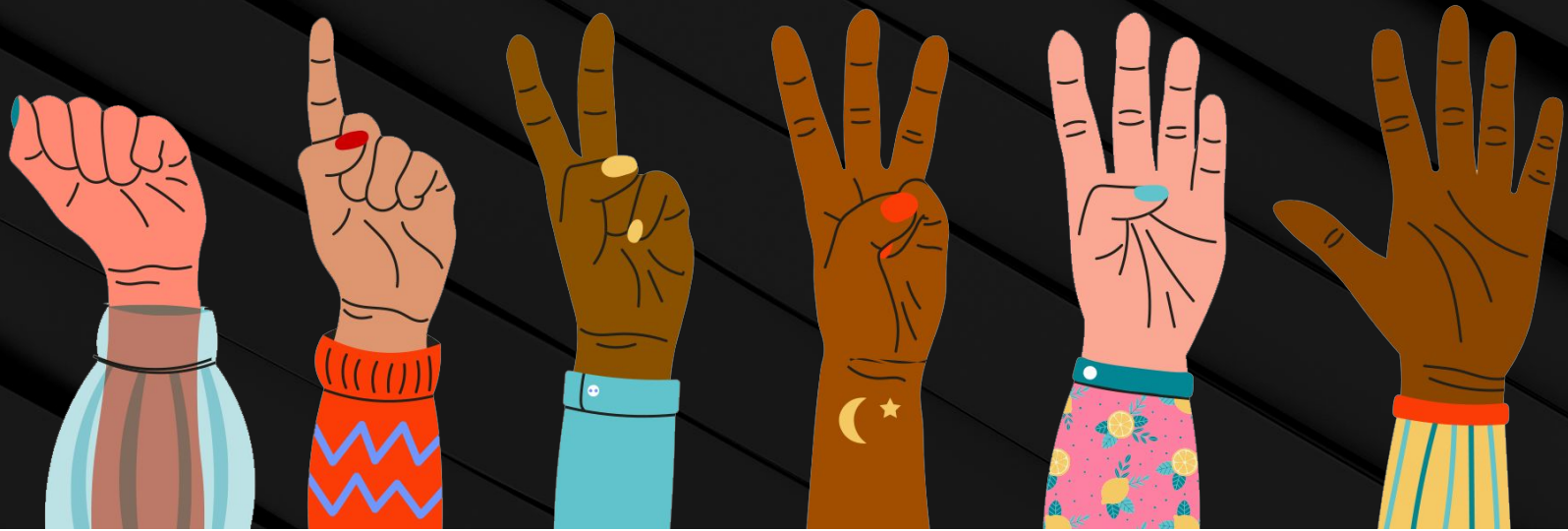
03 Regex Grouping

04 04 Regex with Functions

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?



Sets, Wildcards, and Escaping

Wildcards

Wildcards let us match any characters (letters, digits, whitespace, etc.).
The dot wildcard, `.`, allows us to match 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 allows us to match any character contained within it by using the square brackets.

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

Escaping

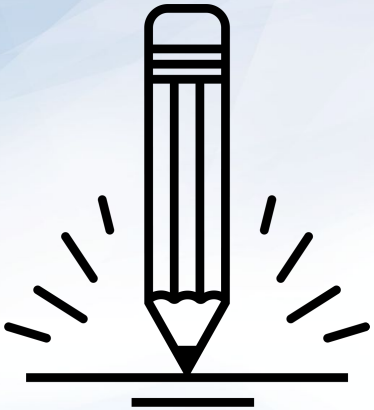
Escaping characters is a way to match characters that are also used in regex, such as the dot, `.`; we do so by placing the delimiter `\` in front of the character that we wish to escape.

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

Instructor Demonstration

Sets, Wildcards, and Escaping

Questions?



Group Activity:
Sets, Wildcards, and Escaping

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

Suggested Time:
15 minutes

Let's Review

Special Characters

Special Characters - ?

The question mark, ?, allows us to 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
```

Special Characters - *

The asterisk, *, allows us to 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
```

Special Characters - ^

The caret, ^, allows us to 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'
```

Special Characters - $

The dollar sign, $, allows us to 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?’
```

Special Characters - |

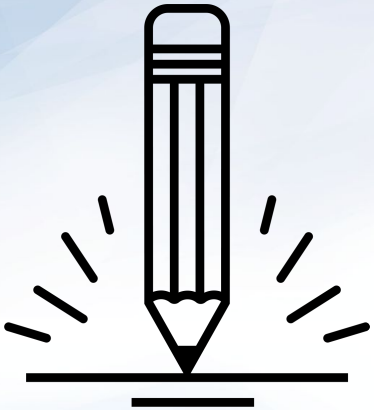
The pipe, |, allows us to put a conditional in our search to match the term either preceding or following it.

```
# 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

Questions?



Group Activity: Special Characters

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

Suggested Time:
15 minutes

Let's Review

Grouping

Additional Regular Expressions

Before we get into grouping, let's introduce a few more regular expression techniques:

`\s`	Matches whitespace
`{4}`	Matches words exactly 4 characters long
`{4,}`	Matches words 4 characters or longer
`{4, 6}`	Matches words 4 to 6 characters long

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

Grouping

Say we want to find all 4 to 6-letter words after the word `Holmes`. We would search with a regex that matches both, but place the word `Holmes` in one group and the six letter word in other. We do this by placing each regular expression 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 results from the previous search would return groups like the following:

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

Instructor Demonstration

Grouping

Questions?

Time to Code

Grouping

Suggested Time:

20 minutes

Let's Review

Regex with Functions

Functional Regex

Suggested Time:

25 minutes

Questions?