Module 3

This Week: Python

This Week: Python

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

- Read and extract data from CSV files.
- Write data to an output file and print the file.
- Recognize Python data types, like integers, floating-point decimal numbers, and strings.
- Declare variables and perform mathematical operations using data types.
- Create and use data structures like lists, tuples, and dictionaries.
- Create and use decision and repetition statements.
- Create and use Boolean and logical operators.

This Week’s Challenge

Using the skills learned throughout the week, complete an audit of election data and provide a written analysis of your findings for the election commission.

Module 3

Today’s Agenda

Today's Agenda

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

01 Decision & Repetition Statements

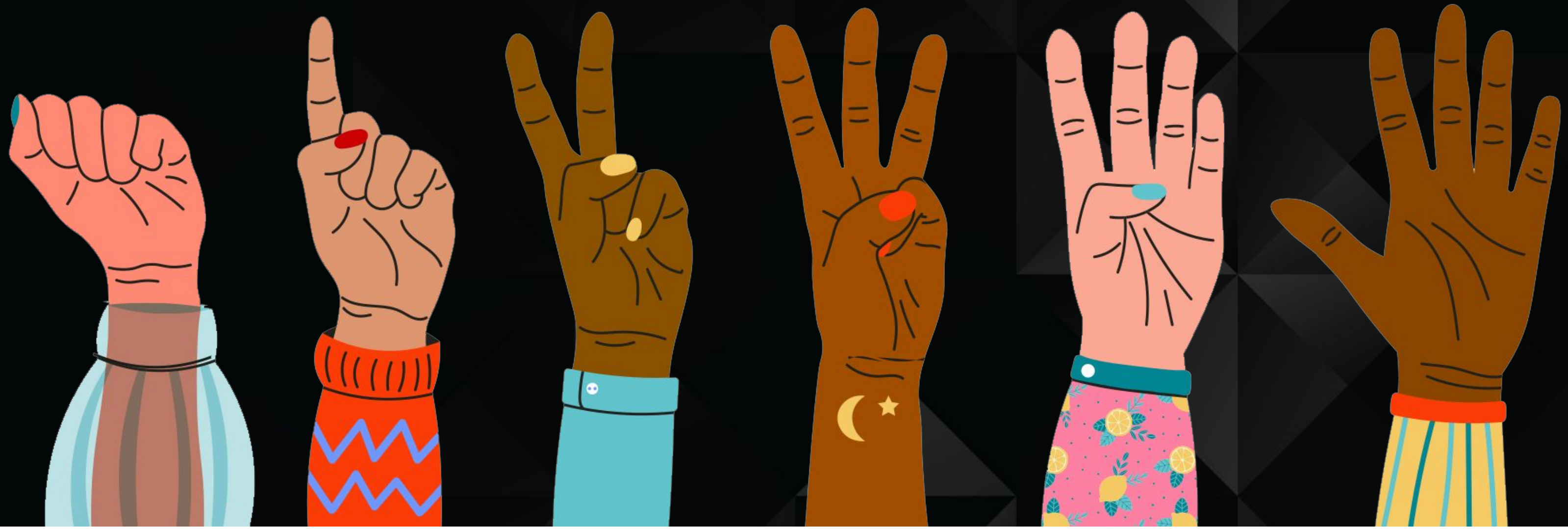
02 Boolean Logic

03 Retrieving & Writing Data to Files

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?



Conditional Conundrum

Recall that **conditional statements** can control the flow of logic based on whether a condition evaluates to `True` or `False`.

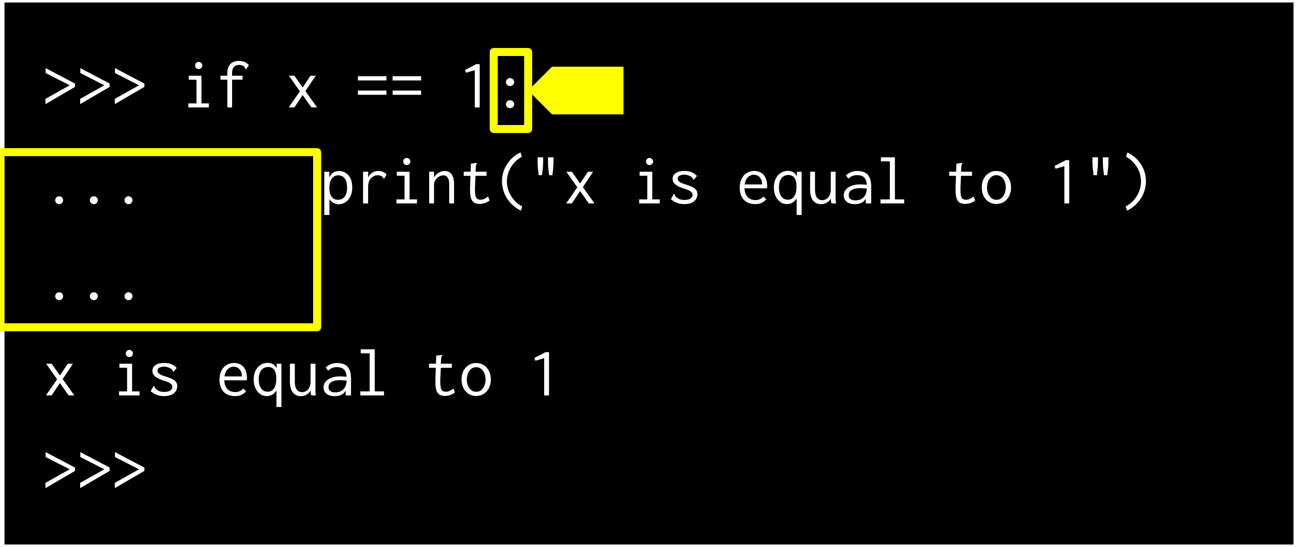
In Python conditional statements are handled by `if`, `elif`, and `else` statements.

Conditionals in Python carry nearly the same logic as in VBA. The primary difference is the syntax and indentation.

Conditionals

Conditional statements are concluded with a colon but all lines after the colon must be indented to be considered a part of that code block. This is because Python reads blocks of code based on indentation.

Hint: count four 'space' stroke on your keyboard or hit 'tab' once



```
>>> if x == 1:
...     print("x is equal to 1")
...
x is equal to 1
>>>
```

Conditionals

Operators like greater than, less than, equal to, and much more can be used to create logic tests for conditionals.

```
>>> x = 10
>>> if x == 1:
...   print('x is equal to 1')
... else:
...   print('x is greater than 1')
```

Conditionals

The condition is equal to uses == while variable assignment uses one equal sign.

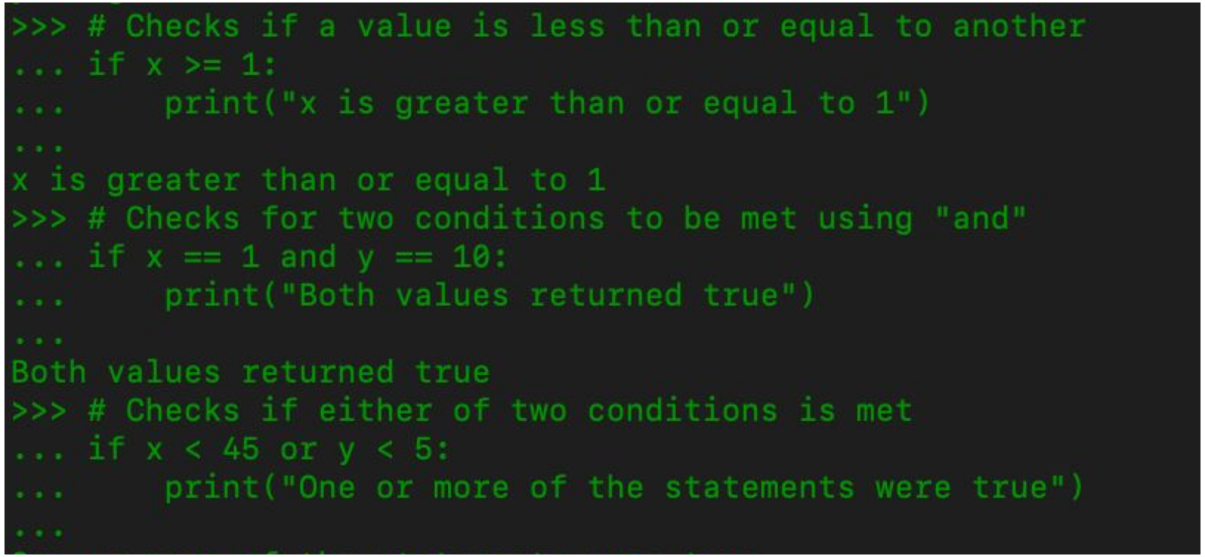
```
>>> x = 10
>>> if x == 1:
...   print('x is equal to 1')
... else:
...   print('x is greater than 1')
```

Logic Tests for Conditionals

>	is greater than
>=	is greater than or equal to
<	is less than
<=	is less than or equal to
==	is equal to
!=	is not equal to

Conditionals

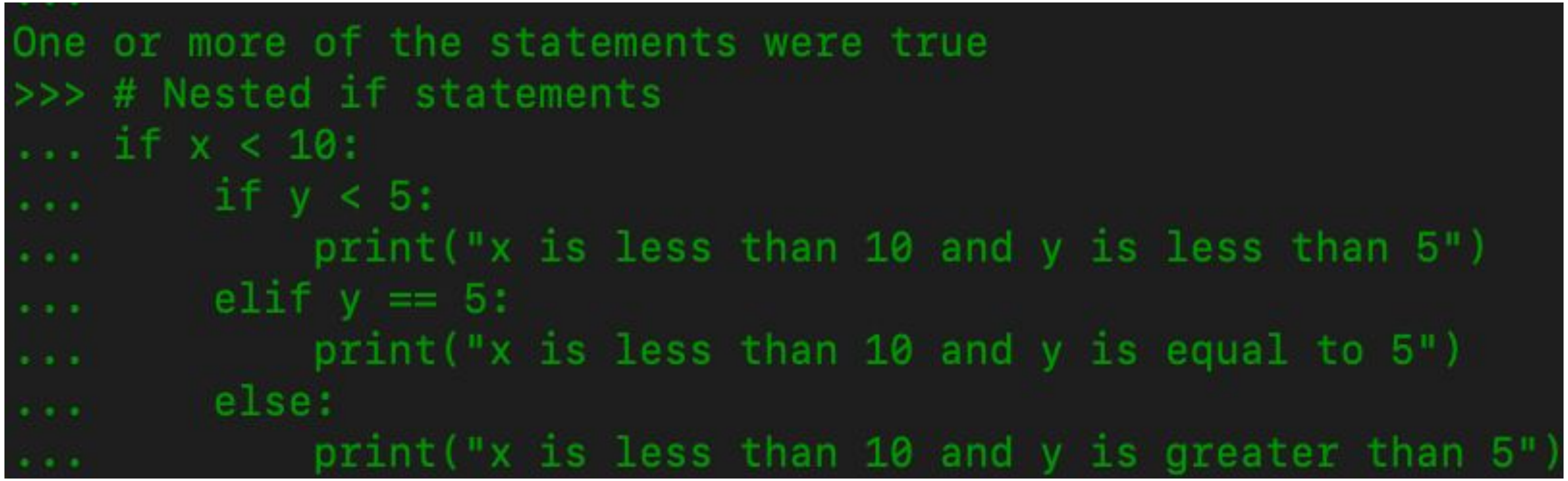
Multiple logic tests can be checked within a single conditional statement. Using the logical operator and requires both tests return as True, while or requires that only one test return as True.



```
>>> # Checks if a value is less than or equal to another
... if x >= 1:
...     print("x is greater than or equal to 1")
...
x is greater than or equal to 1
>>> # Checks for two conditions to be met using "and"
... if x == 1 and y == 10:
...     print("Both values returned true")
...
Both values returned true
>>> # Checks if either of two conditions is met
... if x < 45 or y < 5:
...     print("One or more of the statements were true")
...
```

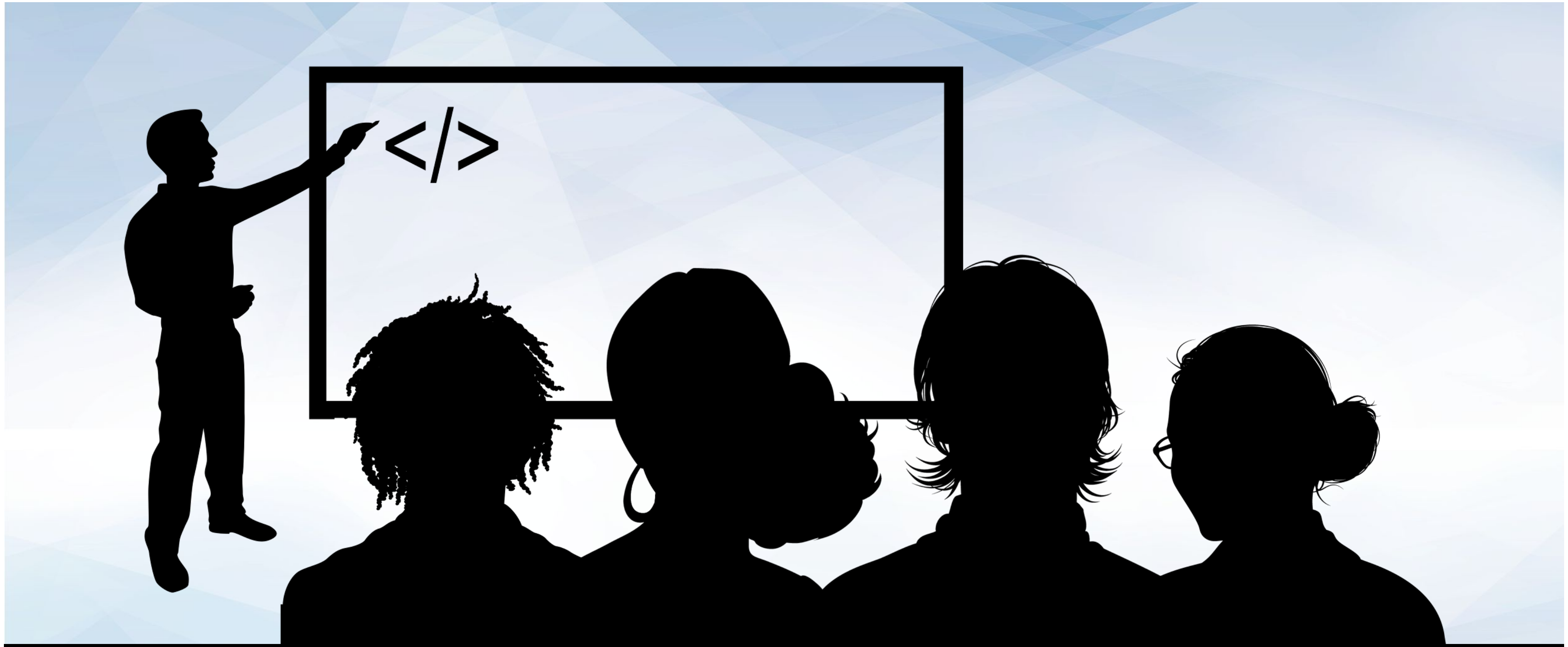
Conditionals

Conditionals can even be nested, allowing programmers to run multiple logic tests based upon whether or not the original logic test returned as True.



```
One or more of the statements were true
>>> # Nested if statements
... if x < 10:
...     if y < 5:
...         print("x is less than 10 and y is less than 5")
...     elif y == 5:
...         print("x is less than 10 and y is equal to 5")
...     else:
...         print("x is less than 10 and y is greater than 5")
```

Questions?



Instructor Demonstration

Conditional Conundrum

Activity: Conditional Conundrum

In this activity, you will need to figure out what lines will be printed to the console with each conditional statement.

Suggested Time:
15 Minutes

Let's Review

Partner Activity:
Rock, Paper, Scissors

In this activity, you will work with a partner to create a simple game of Rock, Paper, Scissors that will run within the console.

Suggested Time:
15 Minutes

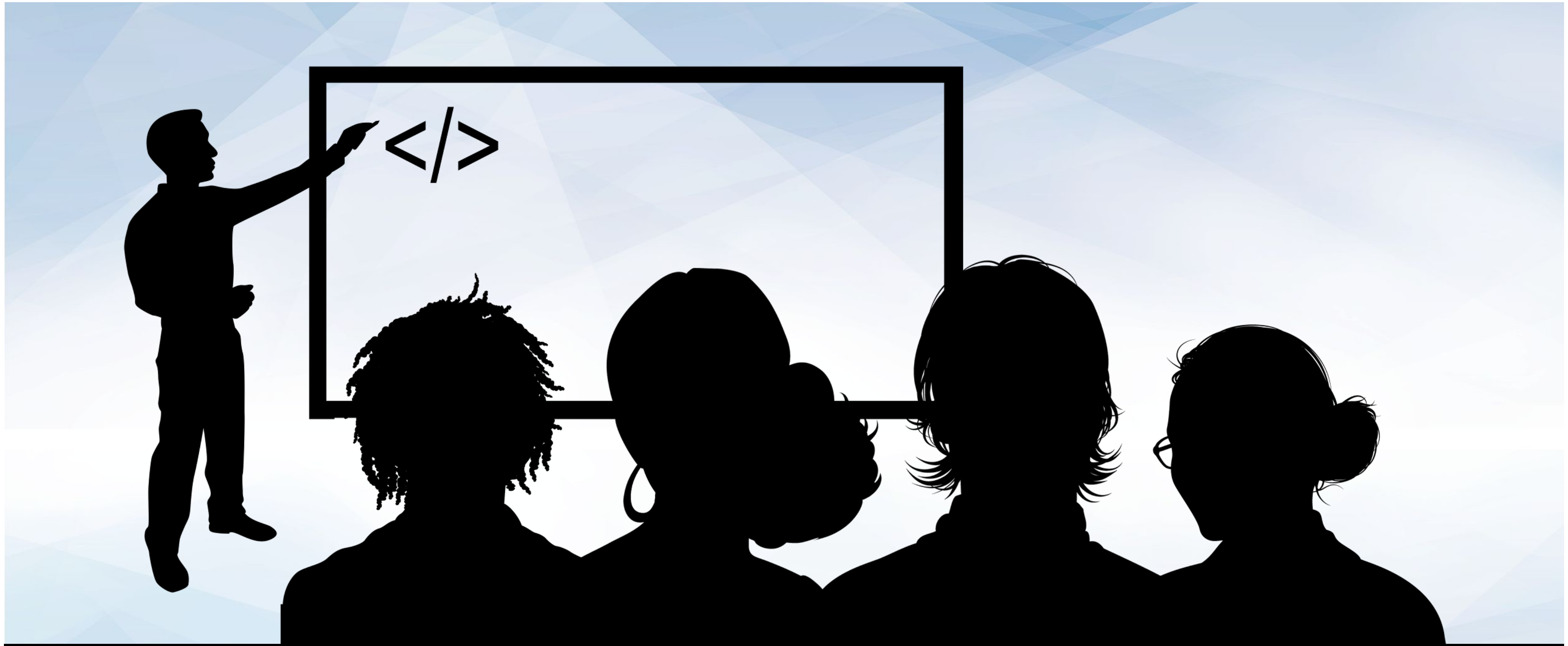
Partner Activity: Rock, Paper, Scissors

Instructions:

- Using the terminal, take an input of r, p, or s, which will stand for (r)ock, (p)aper, or (s)cissors.
- Have the computer randomly pick one of these three choices.
- Compare the user's input to the computer's choice to determine if the user won, lost, or tied.

```
(PythonData) $ python RPS_Solved.py
Let’s play Rock Paper Scissors!
Make your choice: (r)ock, (p)aper, (s)cissors? p
You chose paper. The computer chooses rock.
Congratulations! You won.
```

Let's Review



Instructor Demonstration

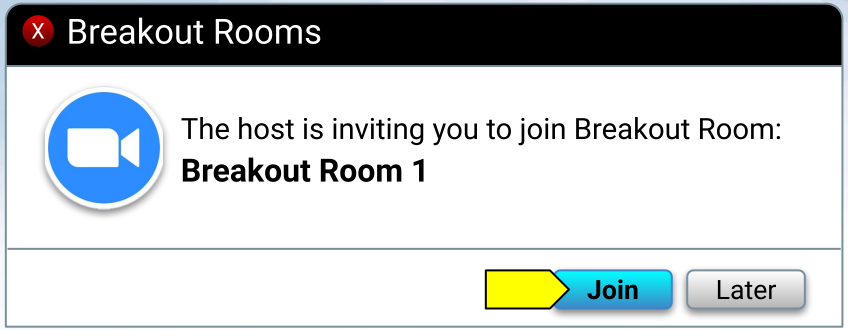
Loops

Questions?

Activity: Number Chain

In this activity, you will take user input and print out a string of numbers.

Suggested Time:
15 Minutes


Breakout Rooms

The host is inviting you to join Breakout Room:
Breakout Room 1

Zoom Breakout Room Activity:

Number Chain

In this activity, you will take user input and print out a string of numbers.

Suggested Time:
15 Minutes

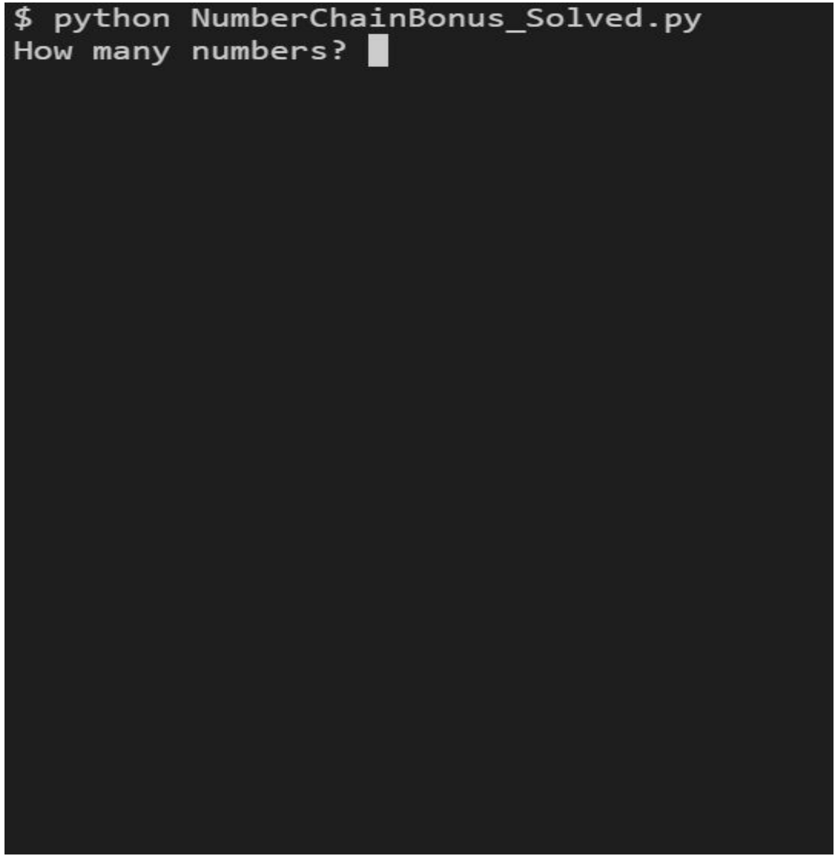
Group Activity: Number Chain

Instructions:

- Using a `while` loop, ask the user ‘How many numbers?’, and then print out a chain of ascending numbers from 0 to the number input.

- After the results have printed, ask the user if they would like to continue. If ‘y’ is entered, keep the chain running by inputting a new number and starting a new count from 0 to the number input. If ‘n’ is entered, exit the application.

- **Bonus:** Rather than just displaying numbers starting at 0, have the numbers begin at the end of the previous chain.



```
$ python NumberChainBonus_Solved.py
How many numbers?
```

Let's Review

Reading In CSV Files

In the data industry, you're more likely to read in **CSV files** than text files.

CSV stands for **Comma Separated Values** and is essentially a table that has been converted into text format with each row and column separated by specified symbols.

Reading in CSV Files

More often than not, each row is located on a new line and each column is separated by a comma. This is why the file type is called Comma Separated Values.

```
First Name,Last Name,SSN
Charlotte,Smith,865414088
Henry,Jones,656077298
Grace,Williams,125486619
Isla,Brown,551953801
Olivia,Wilson,615315318
Ethan,Taylor,907974668
James,Johnson,459599230
Lucas,Walker,996506823
Amelia,Harris,243494429
Eva,Lee,796801997
Noah,Robinson,754601340
```

Python has a module called csv, which pulls in data from external **CSV** files and performs some operations on them.

Reading in CSV Files: os module

The first major piece of code to point out is the importing and usage of the os module. This module allows Python programmers to easily create dynamic paths to external files that function across different operating systems.

```
# First we’ll import the os module
# This will allow us to create file paths across operating systems
import os

csvpath = os.path.join('Resources', 'accounting.csv')
```

Reading in CSV Files: csv reader

This code instead utilises `csv.reader()` to translate the object being opened by Python. It is critical to note the `delimiter=','` parameter being used, as this tells Python that each comma within the CSV should be seen as moving into a new column for a row.

```python
import csv
with open(csvpath, newline='') as csvfile:

    # CSV reader specifies delimiter and variable that holds contents
    csvreader = csv.reader(csvfile, delimiter=',')

    print(csvreader)

    # Each row is read as a row
    for row in csvreader:
        print(row)
```

Reading in CSV Files: csv reader

The code then loops through each row of the CSV and prints out the contents. Each value is being shown as a string and all of the rows are lists.

```
$python ReadCSV.py
['First Name', 'Last Name', 'SSN']
['Charlotte', 'Smith','865414088']
['Henry', 'Jones', '656077298']
['Grace', 'Williams', '125486619']
['Isla', 'Brown', '551953801']
['Olivia', Wilson', '615315318']
['Ethan', 'Taylor', '907974668']
['James', 'Johnson', '459599230']
['Lucas', 'Walker', '996506823']
['Amelia', 'Harris', '243494429']
['Eva', 'Lee', '796801997']
['Noah', 'Robinson', '754601340']
```

Activity: Reading Netflix

In this activity, you'll be provided with a CSV file containing data taken from Netflix to create an application that searches through the data for a specific movie/show and returns the name, rating, and review score for the show.

Suggested Time:
15 Minutes

Activity: Reading Netflix

Instructions:

- Prompt the user to search for a video in the database.
- Search through the `netflix_ratings.csv` to find the user's video.
- If the CSV contains the user's video, then print out the title, what it the movie is rated, and the user ratings. For example: `'Grease is rated PG with a rating of 86'`

Bonus

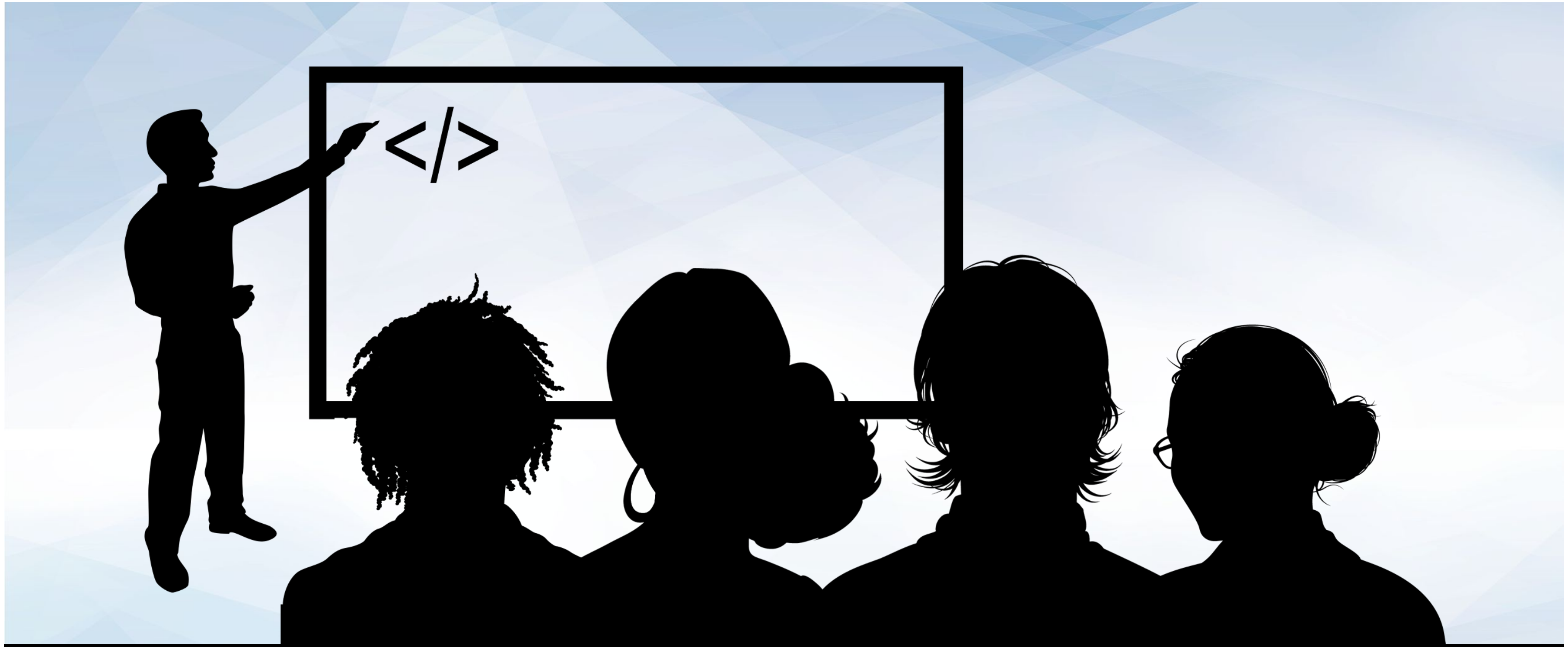
- You may have noticed that there is more than one listing for some videos.
- Edit your code to have the title, the rating, and user rating printed out only once.
- Set a variable to `False` to check if we found the video.
- In the `for` loop, change the variable to confirm that the video is found.
- Insert a `break` statement into the `for` loop to stop the loop when the first movie is found. Check your Slack for documentation.
- If the CSV does not contain the user's video, then print out a message telling them that their video could not be found.

Let's Review

We all know Python can read data from files,

but Python can also write data to a file!

Let’s find out how!



Instructor Demonstration

Writing to a Text File

Activity:
Writing Netflix Data

Suggested Time:
10 minutes

Questions?