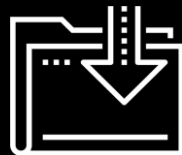# Let's Begin Python

Data Boot Camp
Lesson 3.1

Class Objectives

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

Perform a Python 3 installation.

Navigate thru folders and files via terminal/git-bash.

Create Python scripts and run them in terminal/git-bash.

Understand basic programming concepts in Python.

Without further ado

Without further ado.
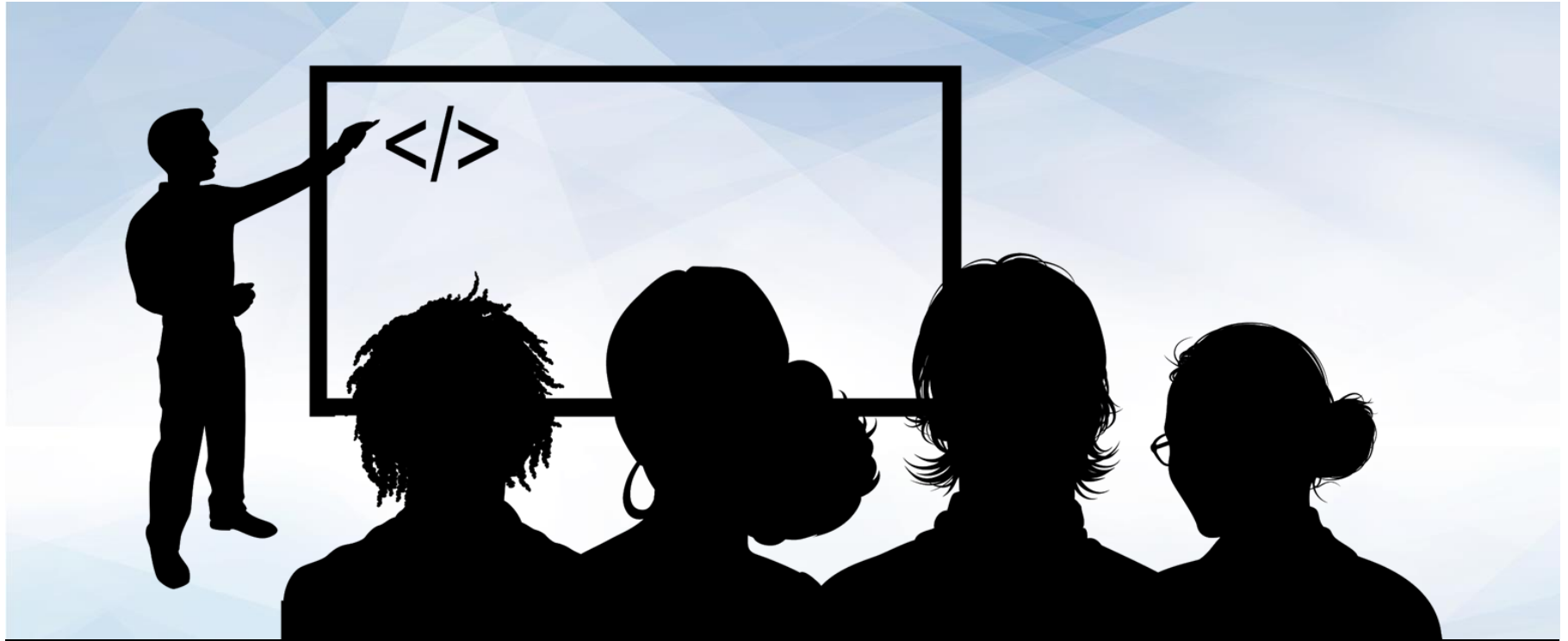
Without further ado..

Without further ado…

Without further ado…
The Mighty Python!!!

Some things to note before we move forward...

- We are diving into a more traditional programming language: Python.

- The fundamental concepts are still the same - the most significant change in this transition from VBA will only be the syntax.

- Check your Slack.

Instructor Demonstration
Terminal

Some basic commands

- `cd` (Changes the directory).
- `cd ~` (Changes to the home directory).
- `cd ..` (Moves up one directory).
- `ls` (Lists files in the folder).
- `pwd` (Shows the current directory – literally "print working directory").
- `mkdir <FOLDERNAME>` (Creates a new directory with the name FOLDERNAME)
- `touch <FILENAME>` (Creates a new file with the name FILENAME).
- `rm <FILENAME>` (Remove: Deletes the file called FILENAME).
- `rm -r <FOLDERNAME>` (Recursive remove: Deletes a folder and its contents; make sure to note the -r).
- `open .` (Opens the current folder on Macs).
- `explorer .` (Opens the current folder on Windows).
- `open <FILENAME>` (Opens a specific file on Macs).
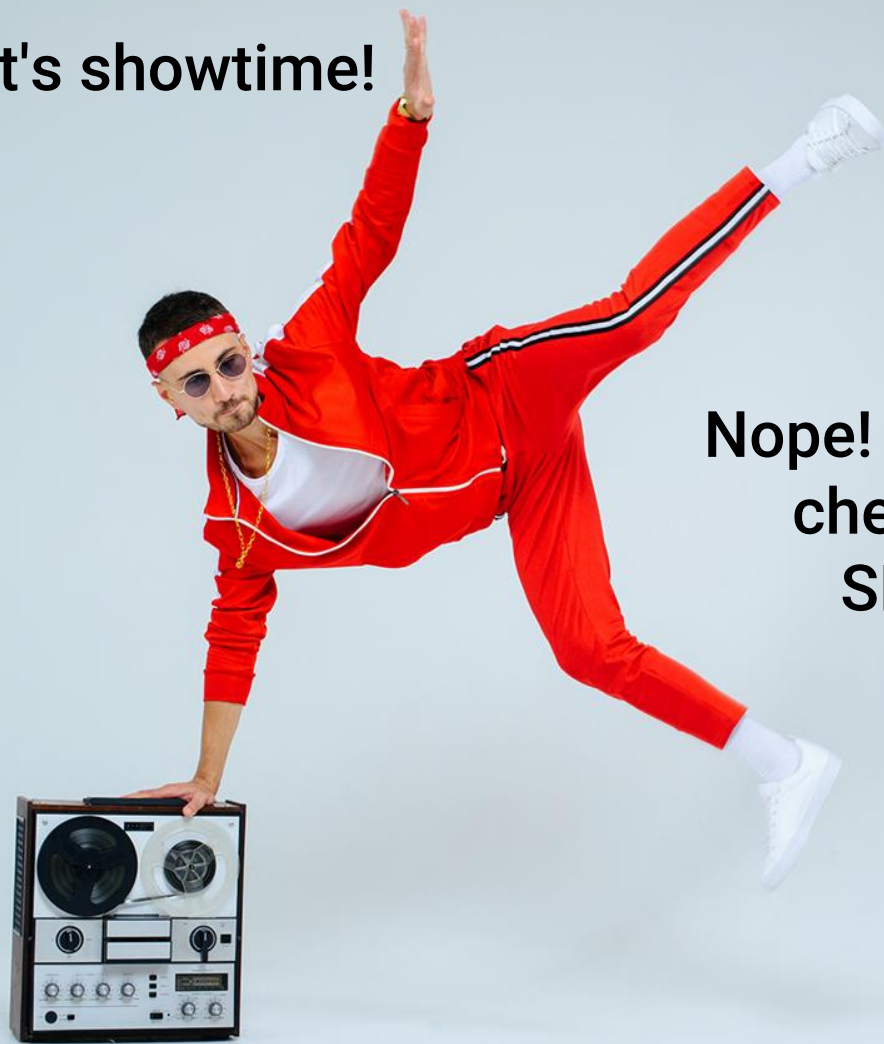- `explorer <FILENAME>` (Opens a specific file on GitBash).

What time is it?

What time is it? It's showtime!

What time is it? It's showtime!

Nope! It's time to check your SLACK!

Common Commands

```
bash-3.2$ mkdir PythonStuff
bash-3.2$ cd PythonStuff
bash-3.2$ touch first_file.py
bash-3.2$ open first_file.py
```

```
bash-3.2$ python first_file.py
bash-3.2$ This is my first_file.py
```

<Time to Code>

Activity: Terminal

In this activity, you will dive into the terminal, create three folders, and a pair of Python files which will print some strings to the console.

Suggested Time:
10 Minutes

Activity: Terminal

Instructions:

- Write and execute the following commands:
    - Create a folder called LearnPython.
    - Navigate into the folder.
    - Inside LearnPython create another folder called Assignment1.
    - Inside Assignment1 create a file called quick_python.py.
    - Add a print statement to quick_python.py.
    - Run quick_python.py.
    - Return to the LearnPython folder.
    - Inside LearnPython create another folder called Assignment2.
    - Inside Assignment2 create a file called quick_python2.py.
    - Add a different print statement to quick_python2.py.
    - Run quick_python2.py.

Time's Up! Let's Review.

Everyone Do: Check Anaconda installation

In this activity, we will check if Anaconda is properly installed.

Suggested Time:
5 Minutes

Everyone Do: Check Anaconda Installation

Check if Anaconda is properly installed

1. Open up terminal

2. Run `conda --version` and hit enter

3. Terminal output should return

`conda 4.x.x`

Everyone Do: Create a Virtual Environment

In this activity, we will create a Virtual Environment with **all right dependencies** for future class activities.

Suggested Time:
10 Minutes

Everyone Do: Create a Virtual Environment

What is a Virtual Environment?

- Virtual environments create an isolated environment for Python projects.

- You may be working on different projects that have different dependencies.

- Different projects might also use different types and versions of libraries.

- This virtual environment will make sure the class has all the right dependencies for future class activities.

Everyone Do: Create a Virtual Environment

Create a Virtual Environment and activate it.

01   Create environment

02   Activate new environment

Run:

```
conda create -n PythonData python=3.6 anaconda
```

Run:

```
source activate PythonData
```

Everyone Do: Create a Virtual Environment

Check python version in the new environment. Exit the environment.

01   Check python version

02   Deactivate new environment

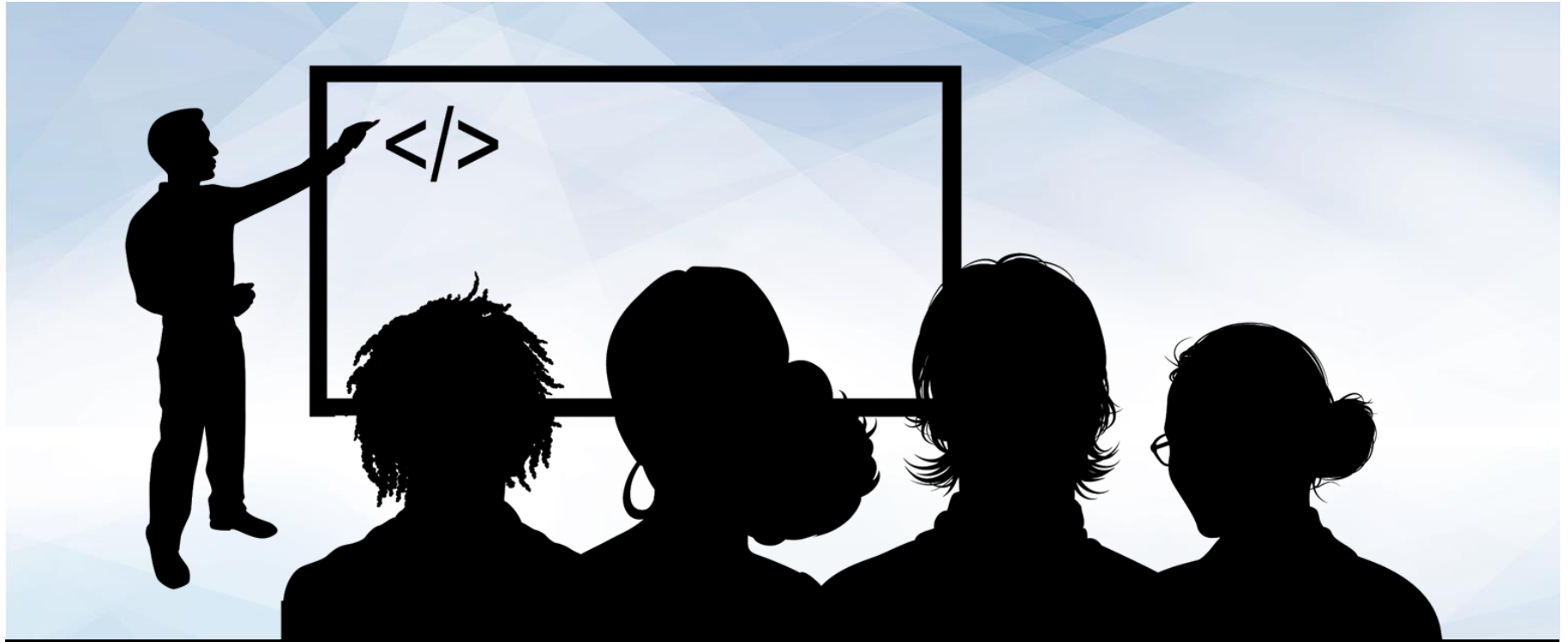**Once in the new environment run the command:**

```
python --version
```

Run:

```
source deactivate
```

Or

```
conda deactivate
```

Instructor Demonstration

Variables

Variables

- Similar to values stored in VBA cells
- In Python, a value is being stored and given a name
- Variables can store different data types like strings, integers and an entirely new data type called booleans which hold True or False values.

```python
# Creates a variable with a string "Frankfurter"
title = "Frankfurter"

# Creates a variable with an integer 80
years = 80

# Creates a variable with the boolean value of True
expert_status = True
```

Print Statements

- We can print statements which include variables, but traditional Python formatting won't concatenate strings with other data types. This means integers and booleans must be casted to strings using the str() function.

```python
# Prints a statement adding the variable
print("Nick is a professional " + title)

# Convert the integer years into a string and prints
print("He has been coding for " + str(years) + " years")

# Converts a boolean into a string and prints
print("Expert status: " + str(expert_status))
```

- Alternatively, the "f-string" method of string interpolation allows strings to be formatted with different data types.

```python
# An f-string accepts all data types without conversion
print(f"Expert status: {expert_status}")
```

Check

Check Your

Check Your Slack

Check Your Slack Again

**Check
Your
Slack
Again**

Check Your Slack Again

<Time to Code>

Activity: Hello Variable World!

In this activity, you will create a simple Python application that uses variables.

Activity: Hello Variable World!

Instructions:

- Create two variables called `name` and `country` that will hold strings.
- Create two variables called `age` and `hourly_wage` that will hold integers.
- Create a variable called `satisfied` which will hold a boolean.
- Create a variable called `daily_wage` that will hold the value of `hourly_wage` multiplied by 8.
- Print out statements using all of the above variables to the console.

```
HelloVariableWorld.py
You live in United States
You are 25 years old
You make 120 per day
Are you satisfied with your current wage? True
```

Time's Up! Let's Review.
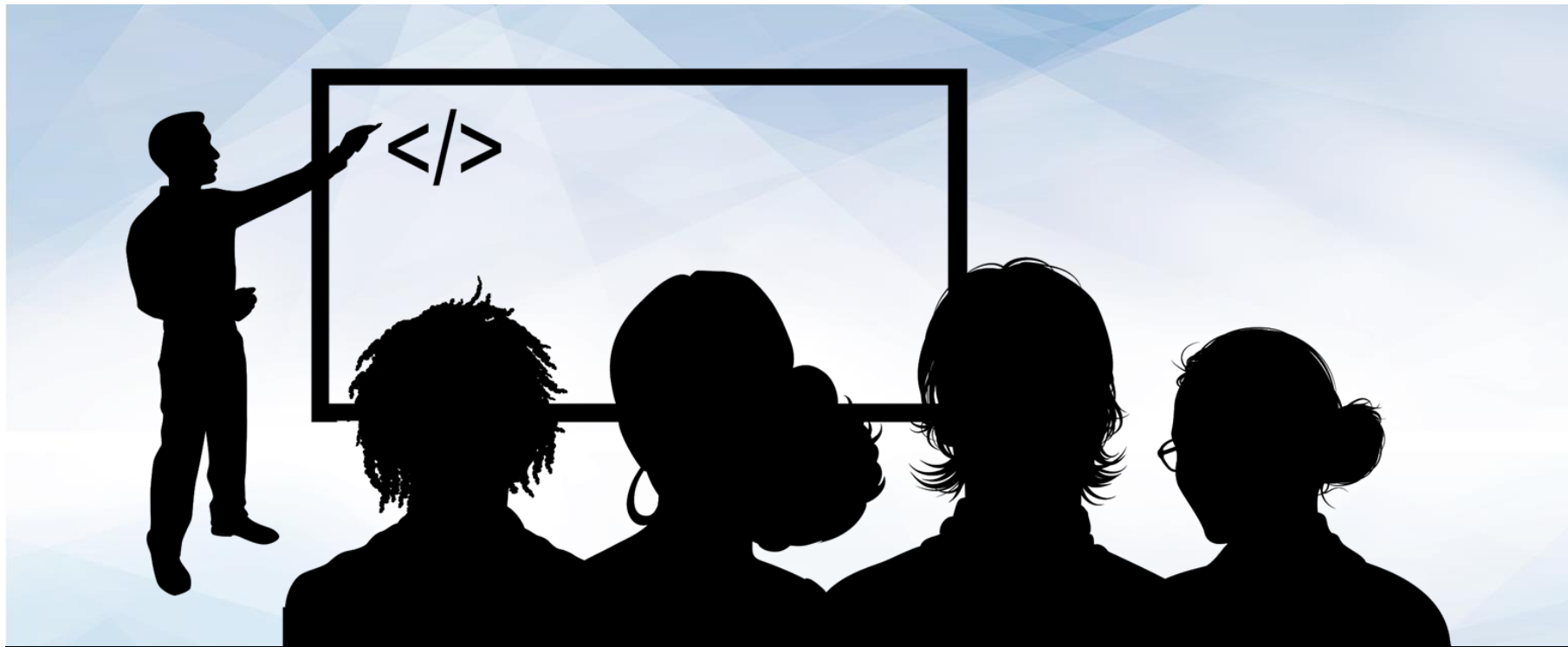
Instructor Demonstration

Inputs and Prompts

Print Statements

- We can print statements which include variables, but traditional Python formatting won't concatenate strings with other data types. This means integers and booleans must be casted to strings using the str() function.

```python
# Prints a statement adding the variable
print("Nick is a professional " + title)

# Convert the integer years into a string and prints
print("He has been coding for " + str(years) + " years")

# Converts a boolean into a string and prints
print("Expert status: " + str(expert_status))
```

- Alternatively, the "f-string" method of string interpolation allows strings to be formatted with different data types. Demonstrate the differences by refactoring the last print statement as an "f-string":

```python
# An f-string accepts all data types without conversion
print(f"Expert status: {expert_status}")
```

Inputs and Prompts

```
(PythonData) $ python inputs.py
What is your name? Gary
How old are you? 33
Is this statement true? yes
My name is Gary
I am 33 years old.
The statement was True
```

<Time to Code>

Activity: Down the input

In this activity, you will work on storing inputs from the command line and run some code based upon the values entered.

Activity: Down to Input

Instructions:

- Create two different variables that will take the input of your first name and your neighbor's first name.
- Create two more inputs that will ask how many months each of you has been coding.
- Finally, display a result with both your names and the total amount of months coding.

```
$ python DownToInput.py
What is your name? Jacob Deming
What is your neighbors name? Bob Smith
How many months have you been coding? 24
How many months has your neighbor been coding? 12
I am Jacob Deming and my neighbor is Bob Smith
Together we have been coding for 36 months!
```

Time's Up! Let's Review.
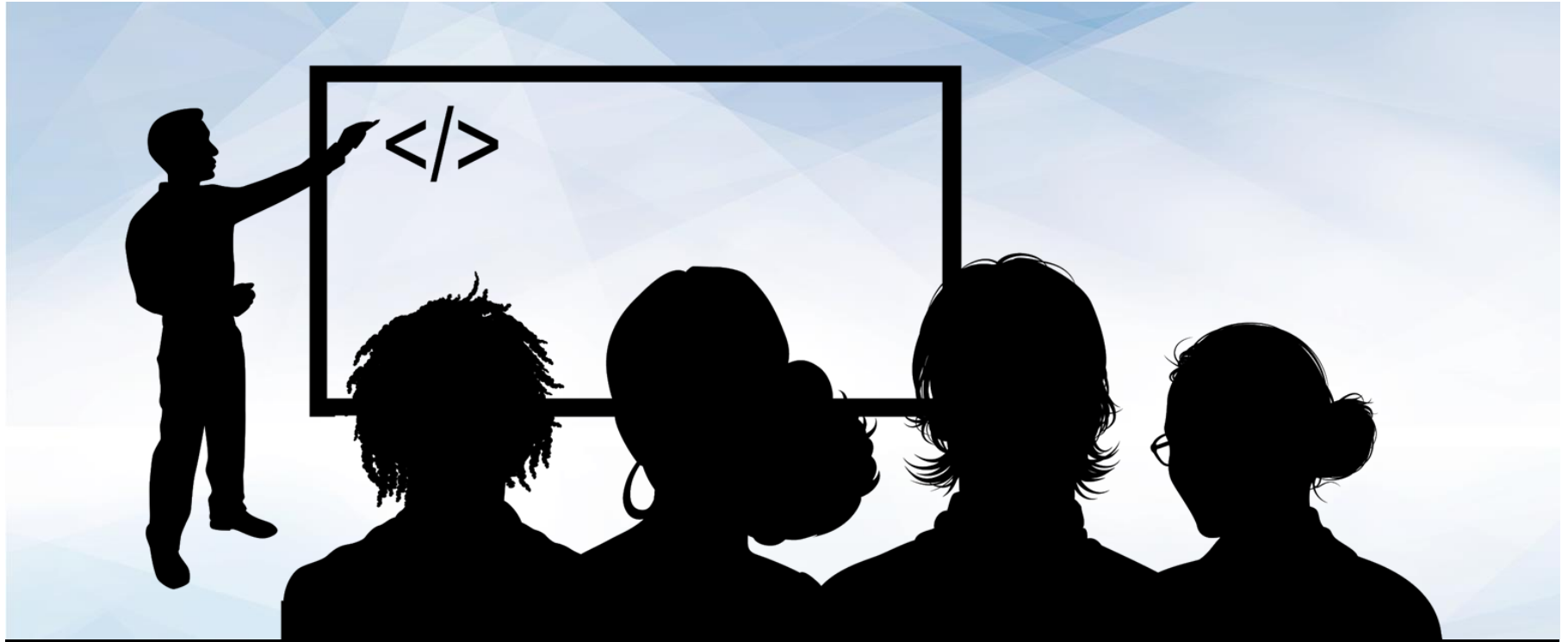
Instructor Demonstration
Conditionals

Conditionals

Some things to keep in mind

- Conditionals in Python carry nearly the same logic as in VBA. The primary difference is the syntax and indentation.
- Python uses `if`, `elif`, and `else` for creating 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.
- All sorts of operators like greater than, less than, equal to, and much more can be used to create logic tests for conditionals.
- The condition "is equal to" uses == while variable assignment uses one equal sign.
- Multiple logic tests can be checked within a single conditional statement. Using the term "and" means all tests must return True while "or" requires that at least one test return as true.
- Conditionals can be nested.

Conditionals

- **Indentation matters in Python!**

```
>>> x = 1
>>> y = 10
>>>
>>> # Look what happens w/o indentation
... if x == 1:
... print('x is equal to 1')
  File "<stdin>", line 3
    print('x is equal to 1')
        ^
IndentationError: expected an indented block
```

```
>>> if x == 1:
```

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

Conditionals

```
>>> # Checks if one value is equal to another
... if x == 1:
...     print("x is equal to 1")
...
x is equal to 1
>>> # Checks if one value is NOT equal to another
... if y != 1:
...     print("y is not equal to 1")
...
y is not equal to 1
>>> # Checks if one value is less than another
... if x < y:
...     print("x is less than y")
...
x is less than y
>>> # Checks if one value is greater than another
... if y > x:
...     print("y is greater than x")
...
y is greater than x
```

```
>>> # 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")
...
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")
```

<Time to Code>

Activity: Conditional Conundrum

In this activity, you will be figuring out what some pre-written conditionals do

Activity: Conditional Conundrum

Instructions:

- Do not run the application.
- Look through the conditionals within the provided code and figure out which lines will be printed to the console.
- Once you've written down your guesses, run the program to check your answers.

- **Bonus:**
  - Create your own series of conditionals to test your fellow students. Once you have completed your puzzle, slack it out to everyone so they can test it.
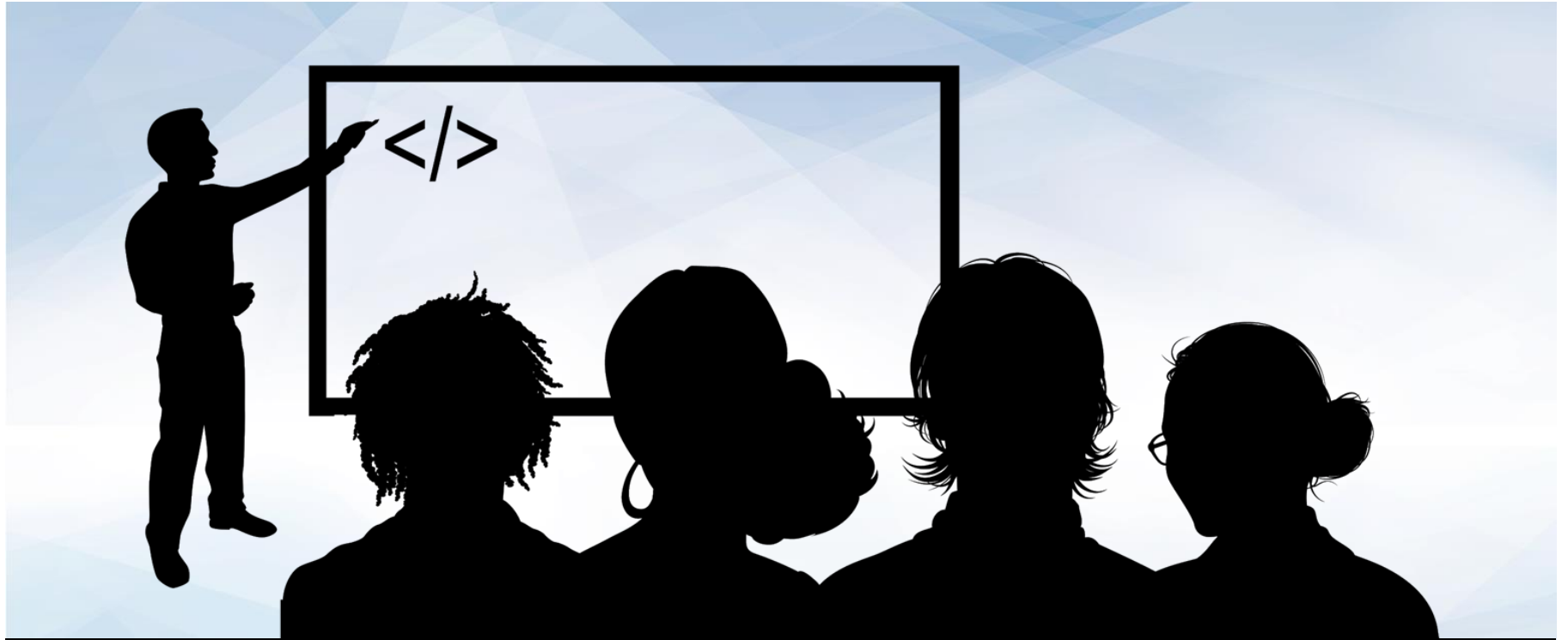
Time's Up! Let's Review.

Sliggety-sliggety-sliggety SLACK!

Instructor Demonstration

Lists

Lists

Some things to keep in mind before we move forward

- Lists are the Python equivalent of arrays in VBA, functioning in much the same way by holding multiple pieces of data within one variable.

- Lists can hold multiple types of data inside of them as well. This means that strings, integers, and boolean values can be stored within a single list.

Lists Methods in Python

Python has a set of built-in methods that you can use on lists

- The `append` method can add elements on to the end of a list.
- The `index` method returns the numeric location of a given value within a list.
- The `len` function returns the length of a list.
- The `remove` method deletes a given value from a list.
- The `pop` method can be used to remove a value by index.

```python
# Create a variable and set it as an List
myList = ["Jacob", 25, "Ahmed", 80]
print(myList)

# Adds an element onto the end of a List
myList.append("Matt")
print(myList)

# Changes a specified element within an List at the given index
myList[3] = 85
print(myList)

# Returns the index of the first object with a matching value
print(myList.index("Matt"))

# Returns the length of the List
print(len(myList))

# Removes a specified object from an List
myList.remove("Matt")
print(myList)

# Removes the object at the index specified
myList.pop(0)
myList.pop(0)
print(myList)
```

Tuples

Tuples are functionally similar to lists in what they can store but are immutable

- While lists in Python can be modified after their creation, tuples can never be modified after their declaration.

- Tuples tend to be more efficient to navigate through than lists and also protect the data stored within from being changed.

```python
# Creates a tuple, a sequence of immutable Python objects that cannot be changed
myTuple = ('Python', 100, 'VBA', False)
print(myTuple)
```

<Time to Code>

Activity: Rock, Paper, Scissors

In this activity, you will be creating a simple game of Rock, Paper, Scissors that will run within the console.

Suggested Time:
## 15 Minutes

Activity: Rock, Paper, Scissors

Instructions:

- Using the terminal, take an input of `r`, `p` or `s` which will stand for rock, paper, and scissors.
- 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 chose rock.
Yay! You won.
```

Time's Up! Let's Review.
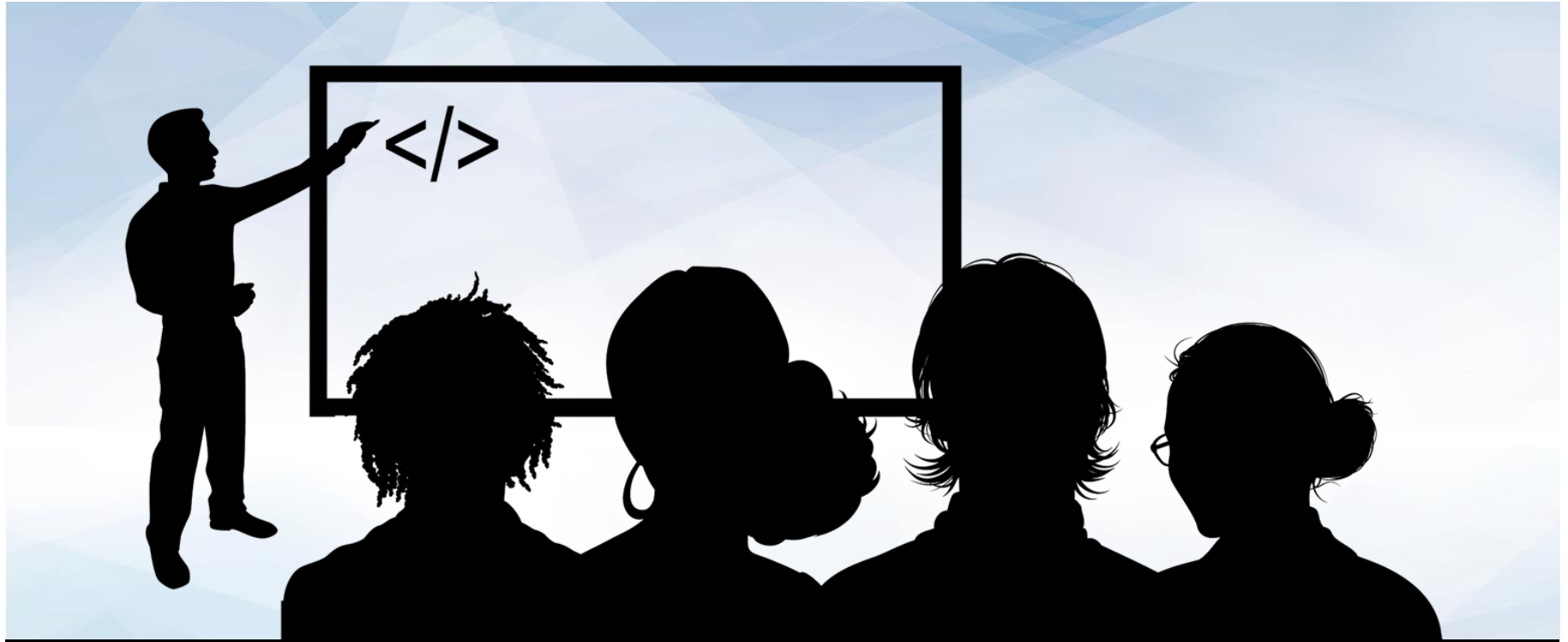
Instructor Demonstration
Loops

Loops

Loops is also a concept we covered in VBA!

- Here, the variable x is created within the loop statement.
- When looping through a range of numbers, Python will halt the loop one number before the final number.
- When looping from 0 to 5, the code will run five times, but x will only ever be printed as 0 through 4.
- When provided with a single number, range() will always start the loop at 0.
- When provided with two numbers, the code will loop from the first number until it reaches one less than the second number.

```python
# Loop through a range of numbers (0 through 4)
for x in range(5):
    print(x)


print("———————————————————————————————————")

# # Loop through a range of numbers (2 through 6)
for x in range(2, 7):
    print(x)


print("———————————————————————————————————")
```

Looping through strings

Python can also loop through all of the letters within a string

- **The syntax is** `for <variable> in <string>:`

```python
# Iterate through letters in a string
word = "Peace"
for letters in word:
    print(letters)

print("--------------------------------------")
```

Looping through lists

Python can also loop through all of the values within a list

- **The syntax is** `for <variable> in <list>:`

```python
# Iterate through a list
zoo = ['cow', 'dog', 'bee', 'zebra']
for animal in zoo:
    print(animal)

print("----------------------------------------")
```

While loops

Just like a for loop but will continue looping for as long as a condition is met

```python
# Loop while a condition is beingv met
run = 'y'

while run == 'y':
    print('Hi!')
    run = input("To run again. Enter 'y'")
```

Activity: Number Chain

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

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?
```

Time's Up! Let's Review.