



The Emergence of Python

Day 1

FinTech
Lesson 2.1



Class Objectives

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



Open a project using JupyterLab.



Pseudocode a task using Python comments.



Create, store, and retrieve data using Python variables.



Control program flow with conditional logic.



Repeat blocks of code with loops.



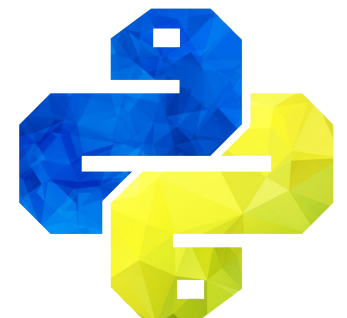


Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Python

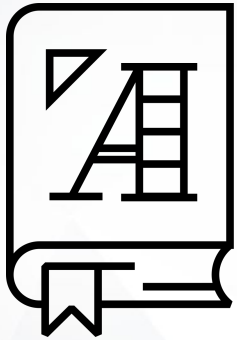
Python offers the following features:

Easy-to-Read Syntax	Modularity	Cross-Platform Capability	Robust Modeling, Financial Analytics, and Data Science Platform
Human-readable syntax that makes it easier to understand and debug applications	<p>A wide variety of standard and advanced libraries that can be easily imported into user applications</p> <p>Many well-known data science and visualization libraries such as NumPy, Pandas, and Plotly</p>	Runtime environments can be deployed on Windows, Mac, Linux	Supports a number of financial analytic and data science libraries such as Pandas, NumPy, Matplotlib, SciPy, and SciKit



The JupyterLab logo is centered on a dark background with a geometric pattern of triangles. The text "JupyterLab" is written in a white, sans-serif font. The background pattern consists of various shades of dark gray and black triangles arranged in a complex, interlocking design.

JupyterLab



JupyterLab is the next version of Jupyter Notebook—an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

JupyterLab

JupyterLab offers the following features:



Enables you to work with documents and activities such as Jupyter Notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner.



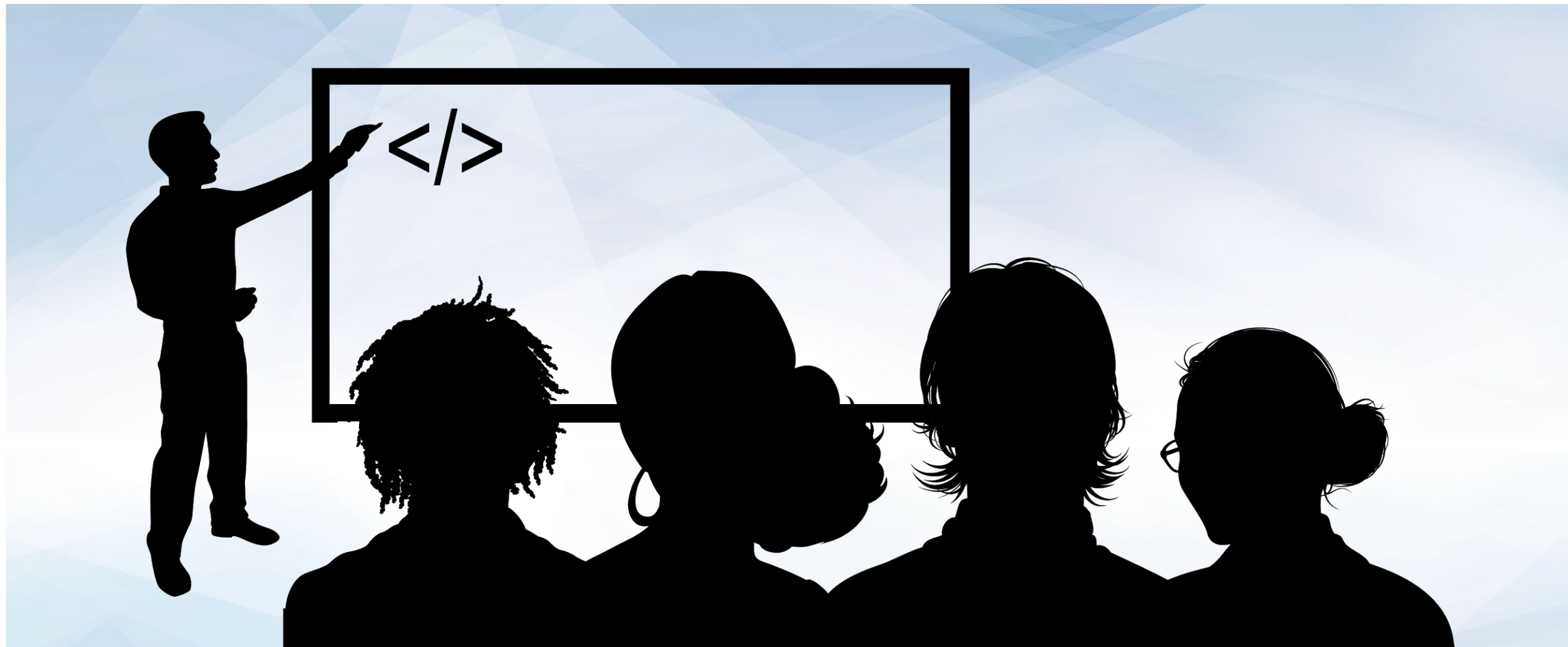
Offers a unified model for viewing and handling data formats: images, CSV, JSON, markdown, PDF, Vega, Vega-Lite, etc.



Arranges multiple documents and activities side by side in the work area using tabs and splitters.



Includes extensions that can customize or enhance any part of JupyterLab, including new themes, file editors, and custom components.



Instructor Demonstration

JupyterLab

JupyterLab Environment Setup

JupyterLab is provided through the Anaconda distribution.

To get started - update your conda environments, activate your Anaconda dev environment, then execute the `jupyter lab` command.

```
conda update conda  
conda update anaconda
```

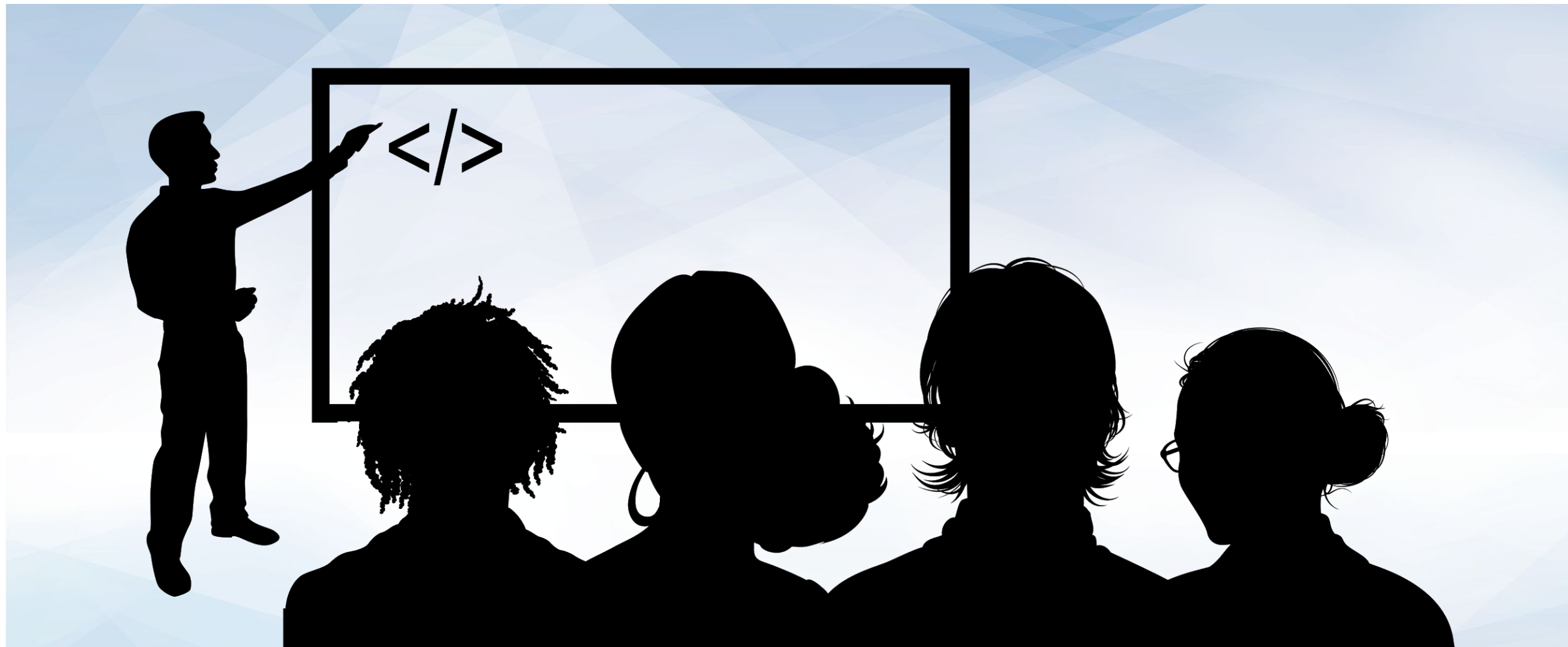
```
conda activate dev
```

```
jupyter lab
```

Environment Issues: They Happen to All of Us

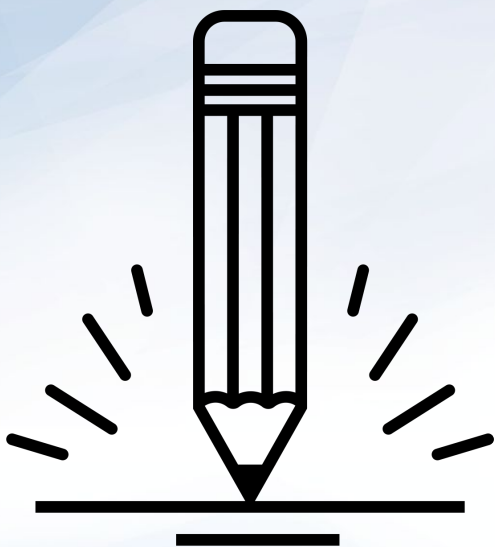
JupyterLab or Anaconda not working? **NO WORRIES**. Raise your hand and a TA will provide assistance.





Instructor Demonstration

Creating Projects with JupyterLab



Activity: Create a JupyterLab Project

In this activity, you will create a JupyterLab project and run a Python `hello world` print statement in a Python notebook.

(Instructions sent via Slack)

Suggested Time:
10 Minutes



Let's Review: Create a JupyterLab Project

01

What is JupyterLab?

02

What are some advantages of JupyterLab?

03

Where are files saved when created in JupyterLab: on the file system or on a separate server?

04

How many activities can be opened in one launcher?

05

What are JupyterLab projects?

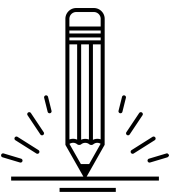
06

How do you create a project in JupyterLab?

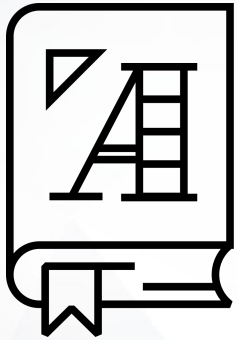
07

What can go into a JupyterLab project?

Review Time: 5 Minutes



Variables



Variables are reserved allocations in memory that can hold defined values.

Variables

Variables have three main operations: **create**, **put**, and **retrieve**.

01

To create a variable, **declare** it.

02

To put a value in a variable, **assign** it.

03

To retrieve a variable, **call** it.

Variables

Commonly used Python data types include:



Number (integer, float, etc.)



String



List



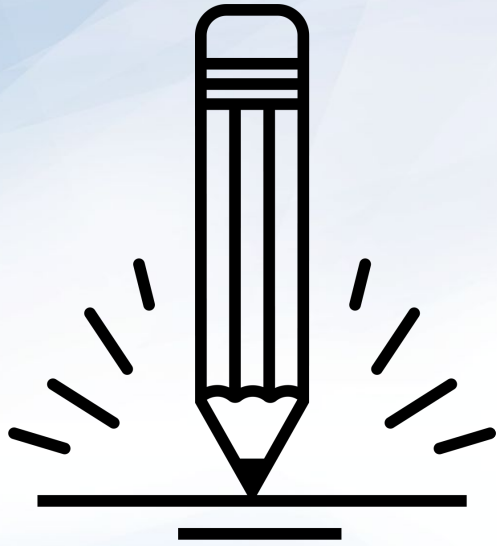
Dictionary



Tuple



Boolean



Activity: Hello Variable World

In this activity, you will learn how to perform calculations and operations using variables.
(Instructions sent via Slack)

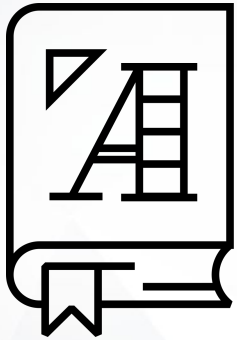
Suggested Time:
10 Minutes





Time's Up! Let's Review.

Conditionals

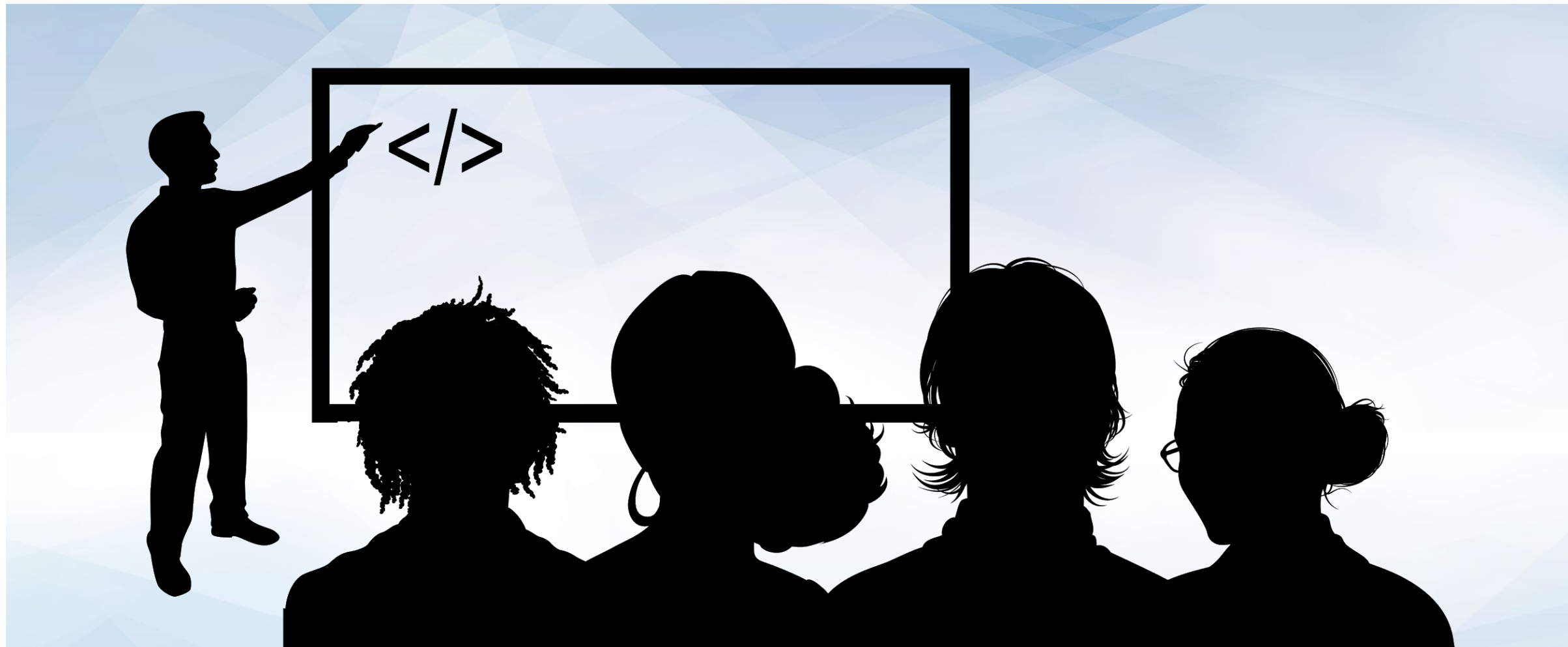


Conditionals are Boolean expressions that evaluate a condition to determine whether it is true or false. The result of the evaluation (true or false) determines the corresponding action.

Conditionals

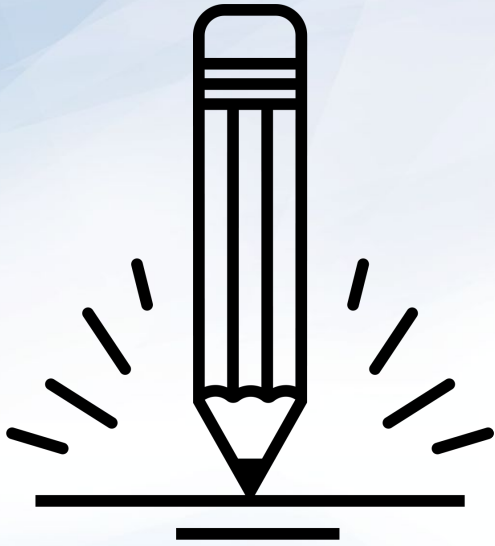
```
if pedestrian:  
    do_not_hit()
```





Instructor Demonstration

Conditionals



Activity: The Conditional Conundrum

In this activity, you will learn how to perform calculations and operations using variables.

(Instructions sent via Slack)

Suggested Time:
15 Minutes



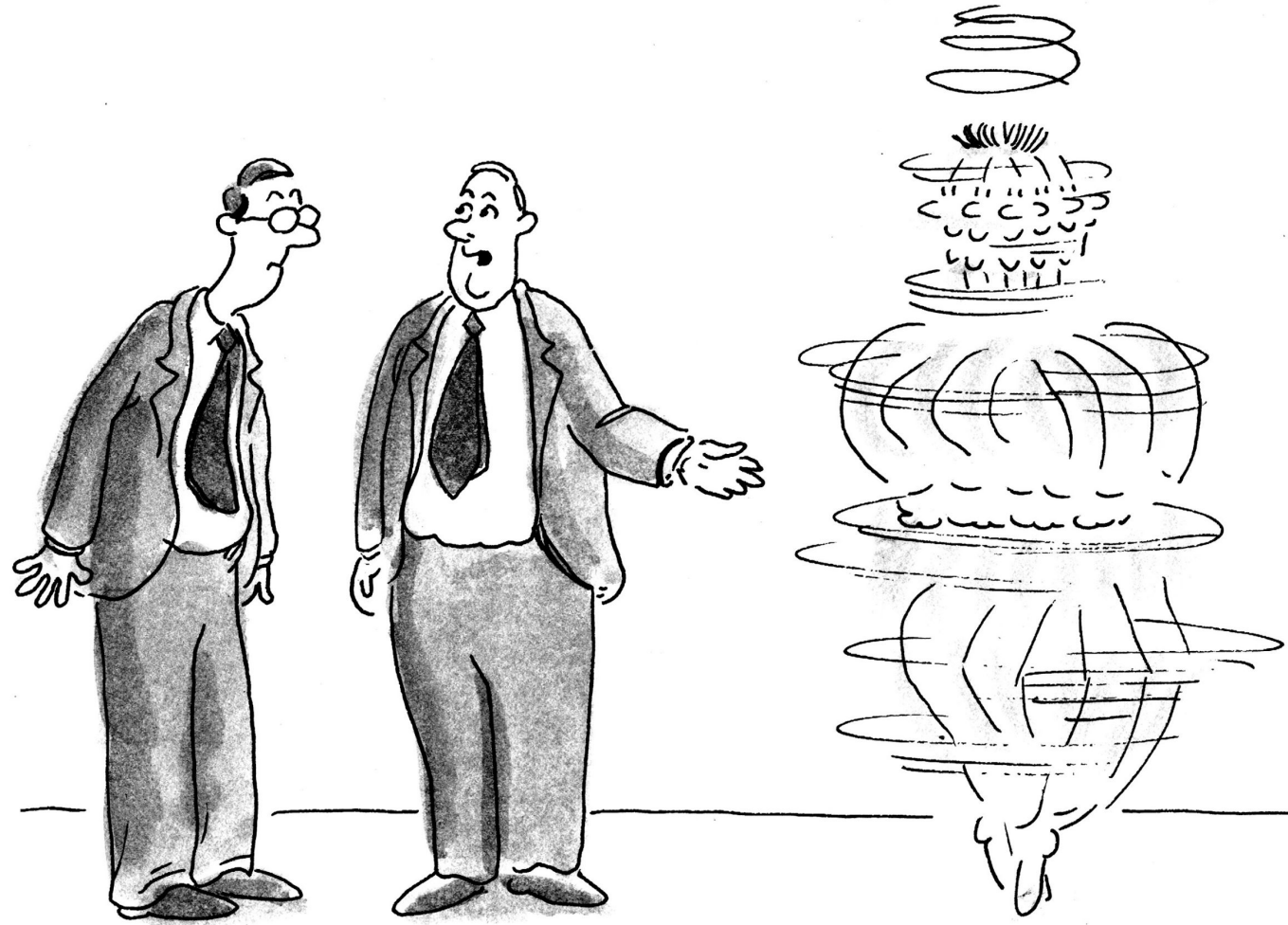


Time's Up! Let's Review.

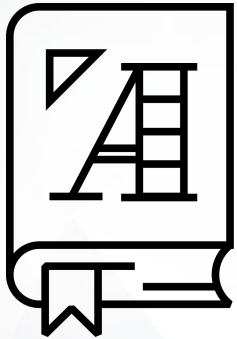
A close-up photograph of a computer keyboard. The central focus is a custom-made white key with rounded corners. On this key, there is a dark blue icon of a coffee cup with three wavy lines above it representing steam. Below the icon, the word "Break" is printed in a dark blue, serif typeface. The key is set against a background of other standard white keyboard keys, including one with a double quote symbol and another with a dash/underscore symbol. The lighting is soft and even, highlighting the texture of the keys and the wood-grain surface of the keyboard base.

Break

For Loops



“Bob is our infinite loop specialist.”



A **for loop** is used for iterating over a sequence such as a list or dictionary.

For Loops

```
# 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 - yes 6!, Up to, but not including, 7)
```

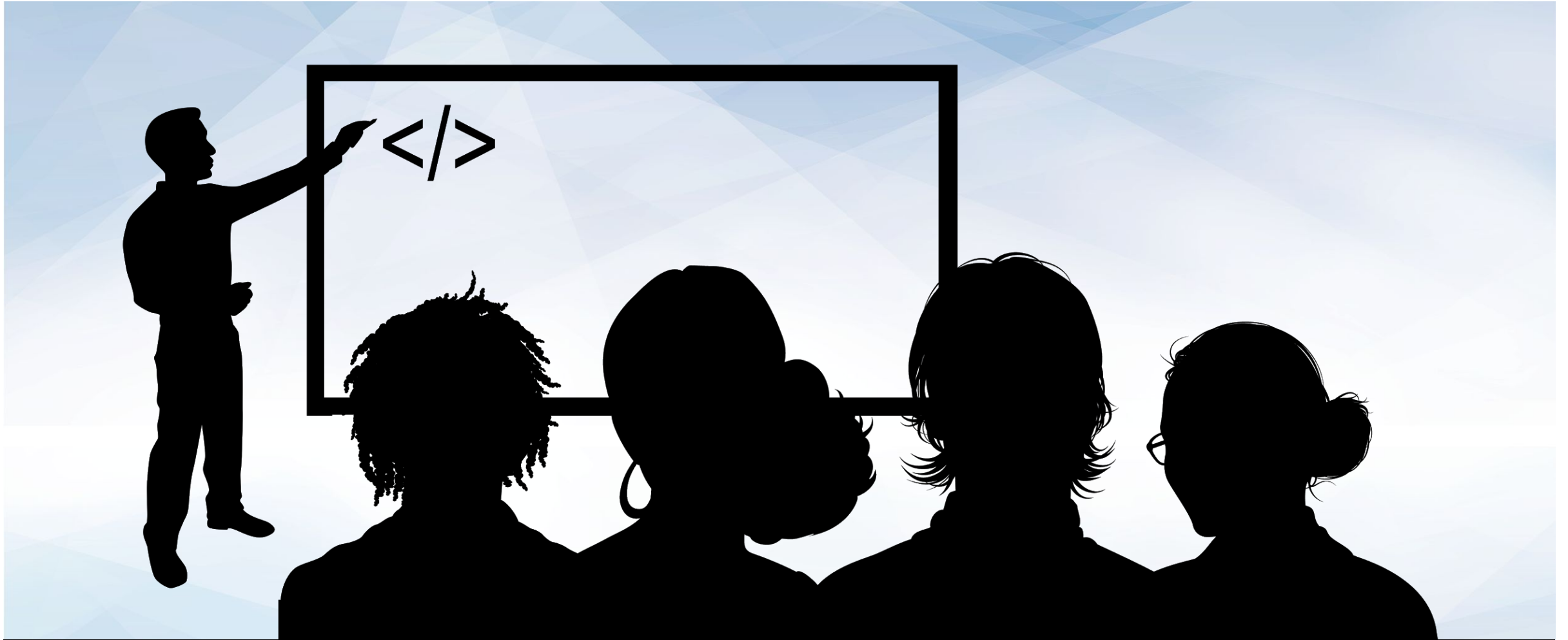
```
for x in range(2, 7):  
    print(x)
```

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

```
# Iterate through letters in a string
```

```
word = "Peace"  
for letters in word:  
    print(letters)
```

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



Instructor Demonstration Loops



Activity: Loop De Loop

In this activity, you will work with for loops in a Python file.

(Instructions sent via Slack)

Suggested Time:
15 Minutes





Time's Up! Let's Review.

Pseudocoding



Pseudocoding is the process of devising the specific requirements and behaviors of an application in human-readable language.

Pseudocoding

Some questions to consider:

What does my application need to do to achieve the result?	What behaviors should my application exhibit?	How should my application perform?
Example: Calculate an average	Example: Login management	The application should be highly available, and data should be able to be queried in real time.
Set a sum value A and a count value B, and return a value that is equal to A / B .	Evaluate a username and password. If they match the security credentials, then log in; else do not log in and display an error.	Replicated cluster nodes such as Hadoop/EMR and MongoDB

Pseudocoding

Pseudocode Example:

```
"""
Pseudocode for a cheerleading program:

1. Initialize "cheer" variable to a
   string to be cheered
2. Create a for loop and iterate through
   each character in "cheer" variable
   2.1. Print each letter to screen with a cheer
3. Print exclamations to screen ("Woohoo!!!")
"""
```

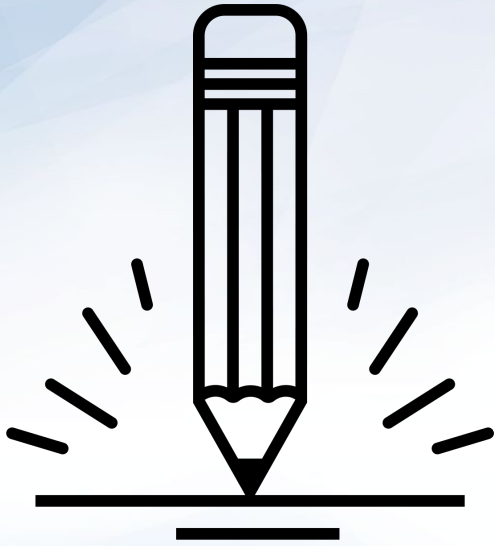
Python Code Example:

```
# Create a variable named cheer
cheer = "Python"

# Below string can be used to add fun
cheer_symbol = "*\0*"
cheer_symbol_2 = "\(^o^)/\(^_^)"

# Loop through a string
for x in cheer:
    #Print each letter with a cheer
    print("Give me a " + x + "!")
    print(x + "!")

# Print excitement to screen
print("\nWhat does that spell?!")
print(cheer + "! \nWoohoo! Go " + cheer + "!")
print(cheer_symbol * 3)
print(cheer_symbol_2)
```



Activity: Conditionally Yours, Part 1

In this activity, you will pseudocode a solution to determine whether a stock should be purchased based on a specific threshold of percent increase or decrease.

(Instructions sent via Slack)

Suggested Time:
10 Minutes



A black silhouette of a person standing on a jagged mountain peak. The person is holding a flag on a pole. A dashed white line runs up the side of the mountain, suggesting a path or a stock price trend.

Challenge:

Conditionally Yours, Part 2

In this activity, you will leverage the pseudocode created in the last activity to develop a program that recommends a stock to be purchased based on a specific threshold of percent increase or decrease.

(Instructions sent via Slack)

Suggested Time:
15 Minutes



Let's Review: Pseudocoding



How did pseudocoding improve or hinder your coding process?



What are some best practices for pseudocoding?



True or false: Pseudocode should include lines of code.



How do you identify a dependent statement in pseudocode?

Review Time: 5 Minutes



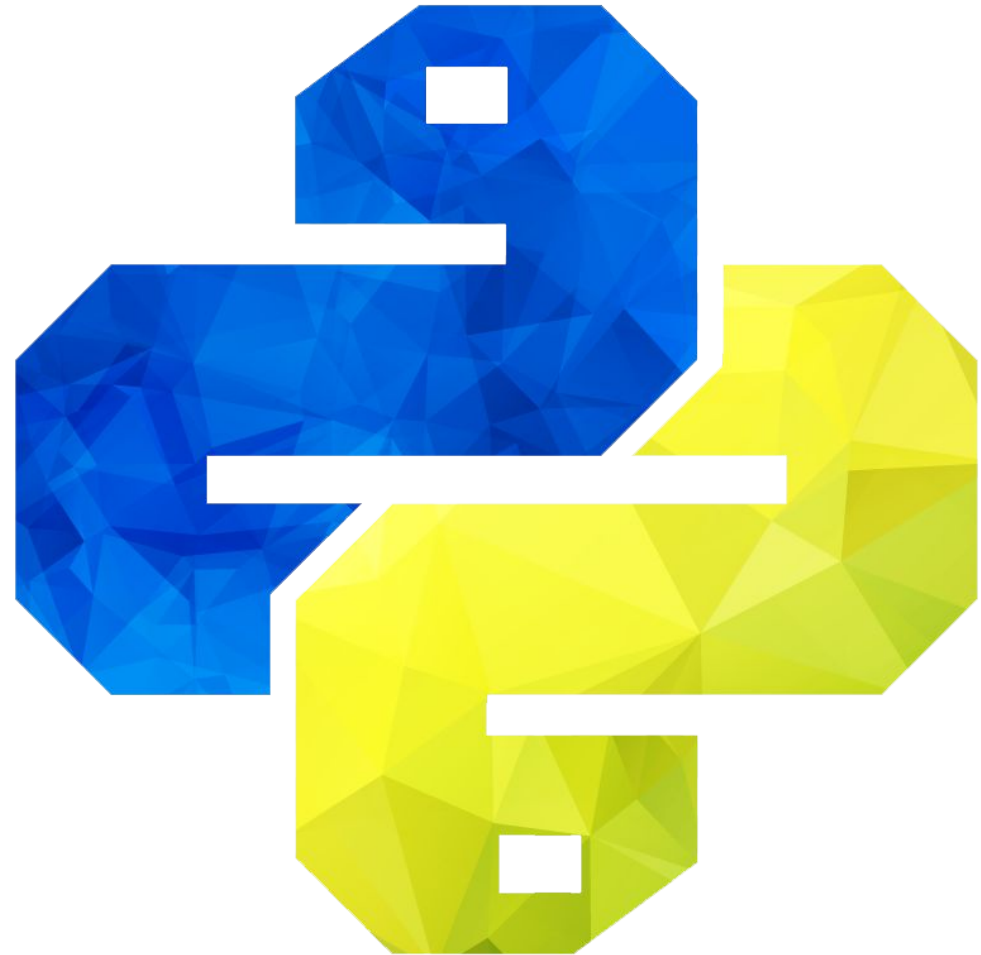
Preview Homework

Decompress

Give Yourself a Round of Applause!

Rejoice! You went to infinity and beyond today!
Great, great, great work surviving today's
mental battles with Python.

And it seems you've gained some Python
prowess since the beginning of class—your
fingers are looking kind of buff.



Today's Class

Here's a high-level view breakdown of what you learned today:



How to work with JupyterLab projects



Python syntax and keywords



Variables



Conditionals



For loops



Pseudocoding