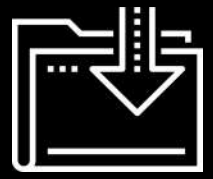
A Deeper Dive Into Python

Data Boot Camp

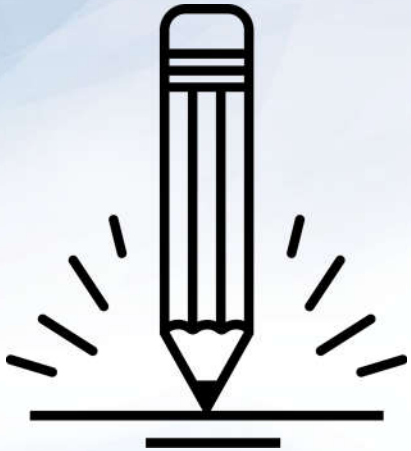
Lesson 3.3



Class Objectives

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

- Create and use Python dictionaries.
- Read in data from a dictionary.
- Use list comprehensions.
- Write and reuse Python functions.
- Use coding logic and reasoning.
- Add, commit, and push code to GitHub from the command line.



Activity: Cereal Cleaner

In this activity, you will create an application that reads in cereal data from a CSV file and then prints only those cereals that contain 5 or more grams of fiber.

Suggested Time:
20 Minutes

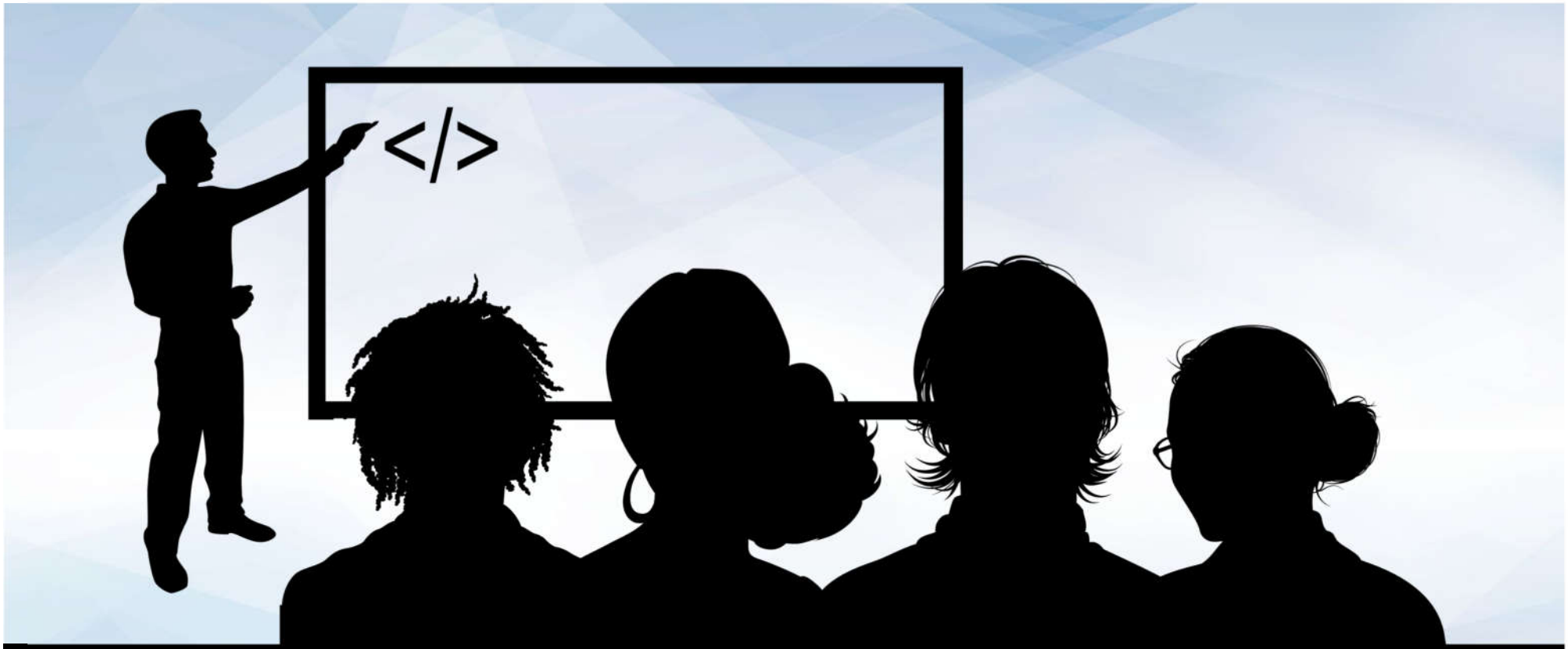
Instructions

Activity: Cereal Cleaner

- Read through `cereal.csv`, find the cereals that contain five grams of fiber or more, and print the data from those rows to the terminal.
- **Hints:**

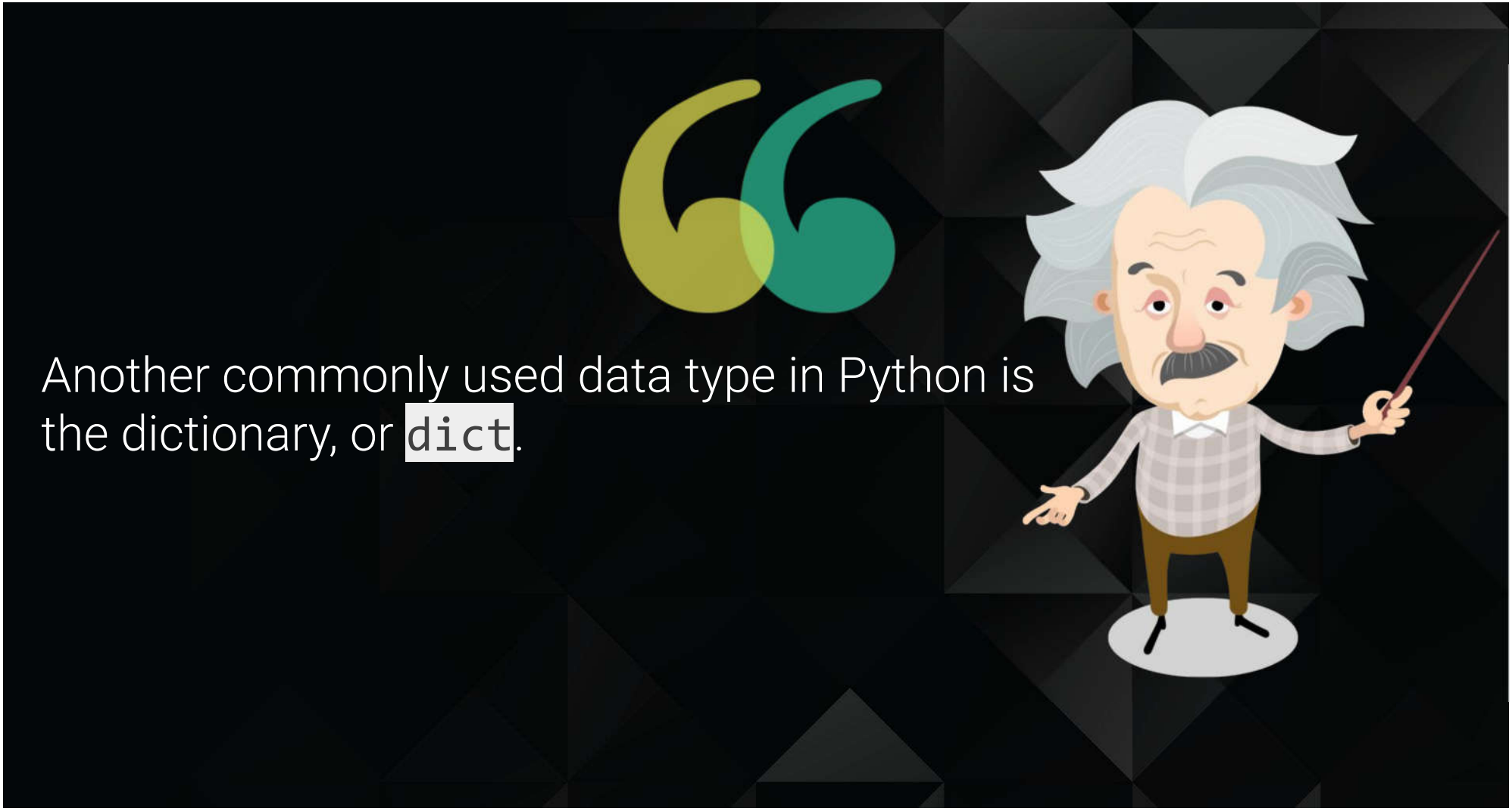
  - Every value within the CSV is stored as a string, and certain values have a decimal. This means that they will have to be cast to be used.
  - `csv.reader` begins reading the CSV file from the first row. `next(csv_reader, None)` will skip the header row.
  - Integers are whole numbers and, as such, cannot contain decimals. Decimal numbers will have to be cast as a `float` or `double`.
- **Bonus:** Try the activity again, but this time use `cereal_bonus`, which does not include a header.

Time's Up! Let's Review.



Instructor Demonstration

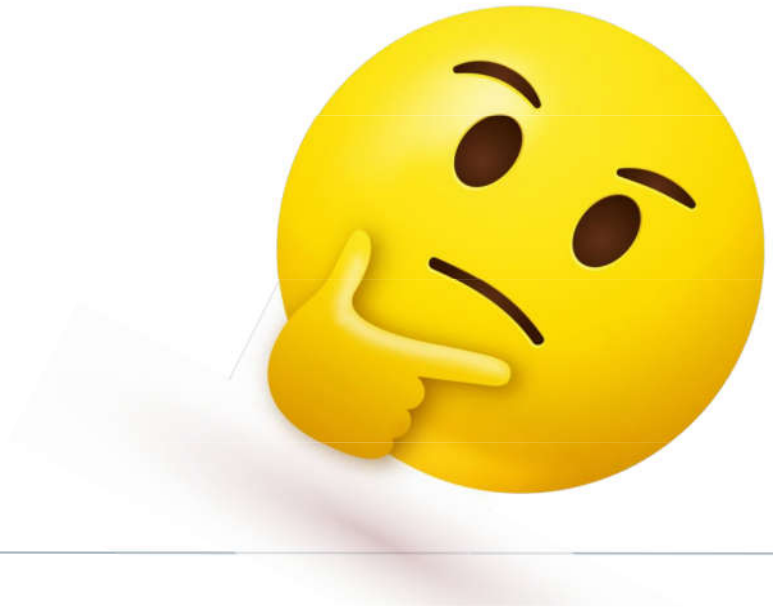
Dictionaries


Another commonly used data type in Python is the dictionary, or dict.

Dictionaries

A dictionary is an object that stores a collection of data.

- Like lists and tuples, dictionaries can contain multiple values and data types.
- However, unlike lists and tuples, dictionaries store data in key-value pairs. The key in a dictionary is a string that can be referenced to collect an associated value.
- To use the example of a physical dictionary, the words in the dictionary would be considered the keys, and the definitions of those words would be the values.



Let's Code...

Dictionaries

- To initialize or create an empty dictionary, we use the following syntax, `actors ={}`.

```
# Create a dictionary to hold the actor's names.
actors = {}
```

- You can also create a dictionary with the built-in Python `dict()` function, `actors = dict()`.

```
# Create a dictionary using the built-in function.
actors = dict()
```

Dictionaries

- Values can be added to dictionaries at declaration by creating a key that is stored within a string, following it with a colon, and then placing the desired value after the colon.
- To reference a value within a dictionary, we simply call the dictionary and follow it up with a pair of brackets containing the key for the desired value.

Dictionaries

Values can also be added to dictionaries by placing the key within single or double quotes inside brackets, and then assigning the key a value. Then, values can be changed or overwritten by assigning the key a new value.

```
>>> # Add an actor to the dictionary with the key "name" and the value "Denzel Washington".
... 
```

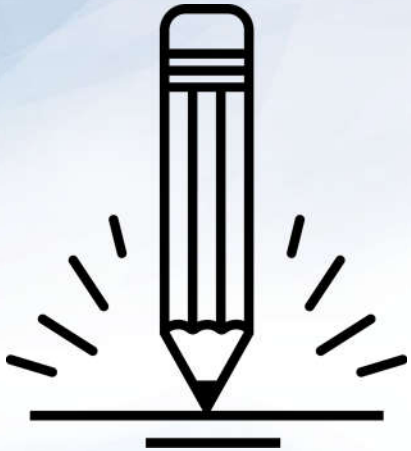
Dictionaries

Dictionaries can hold multiple pieces of information by following up each key-value pairing with a comma and then another key-value pair.

- Keys are immutable, or fixed, objects, like integers, floating-point decimals, or strings. Keys cannot be lists or any other type of changeable object.
- Values in a dictionary can be objects of any type: integers, floating-point decimals, strings, Boolean values, `datetime` values, and lists.

Dictionaries

Dictionaries can also contain other dictionaries. To access the values inside nested dictionaries, simply add another key to the reference.



Activity: Hobby Book

In this activity, you will create and access dictionaries that are based on your own hobbies.

Suggested Time:
15 Minutes

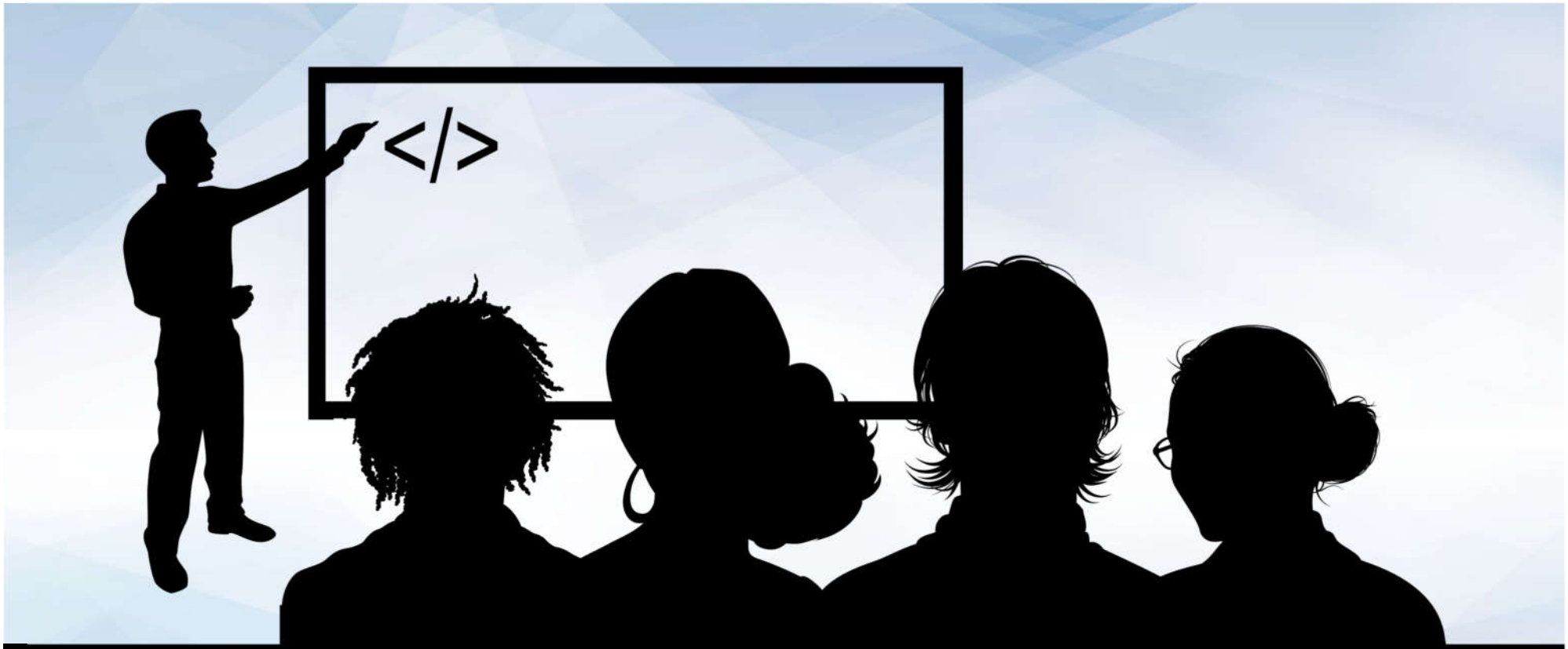
Activity: Hobby Book

Instructions:

1. Create a dictionary that will store the following:
   - Your name
   - Your age
   - A list of a few of your hobbies
   - A dictionary that includes a few days and the time you typically wake up on those days

2. Print out your name, how many hobbies you have, and a time you typically wake up during the week.

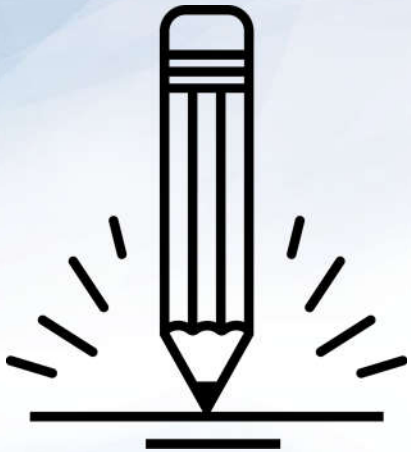
Time's Up! Let's Review.



Instructor Demonstration

List Comprehensions

Get Ready for Some Live Coding!



Activity: List Comprehensions

In this activity, you will use list comprehensions to compose a wedding invitation to send to every name on your mailing list.

Suggested Time:
10 Minutes

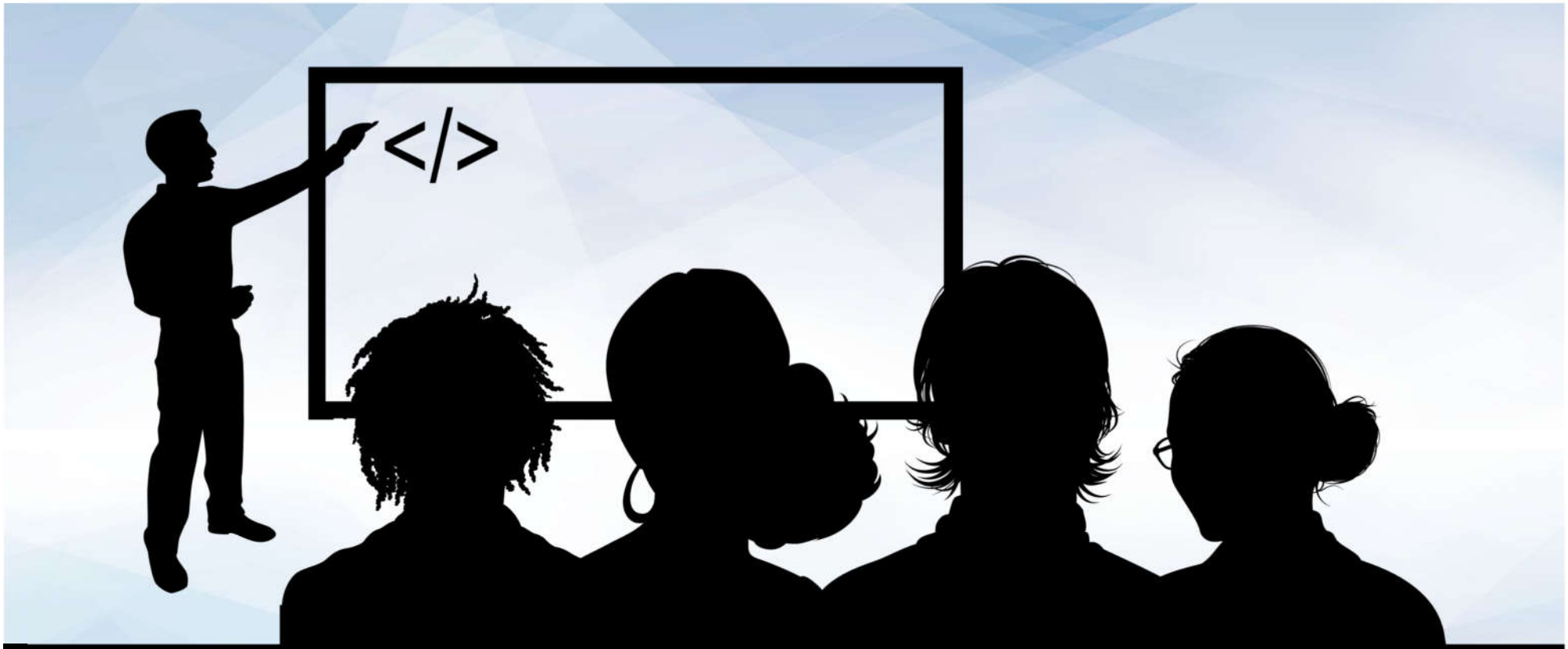
Instructions

Activity: List Comprehensions

- Open the file called `comprehension.py`.
- Create a list that prompts the user for the names of five people that they know.
- Run the provided program. Note that nothing forces you to write the name "properly"—meaning, as "Jane" instead of "jAnE". You will use list comprehensions to fix this.
    - First, use list comprehensions to create a new list that contains the lowercase version of each of the names your user provided.
    - Then, use list comprehensions to create a new list that contains the title-case versions of each of the names in your lowercase list.

Hint: Check out the documentation for the title method. 

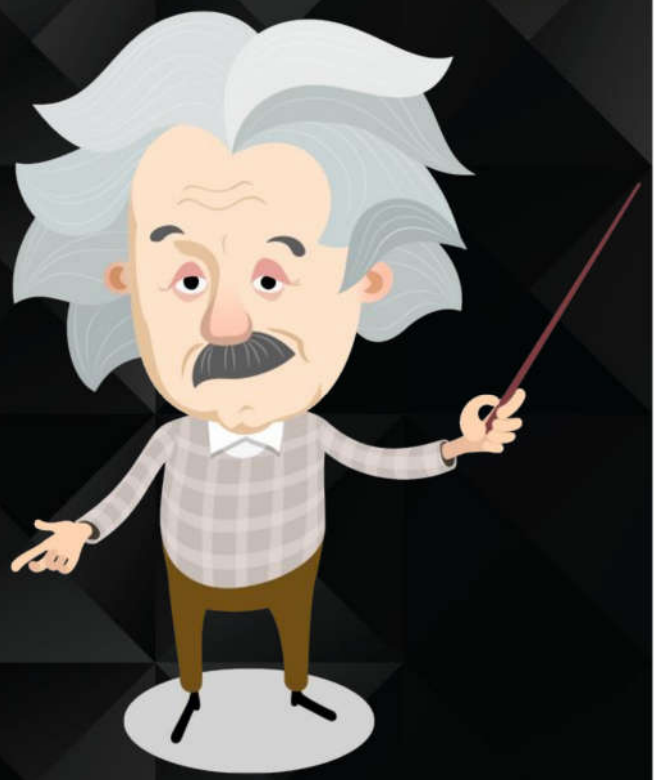
Time's Up! Let's Review.



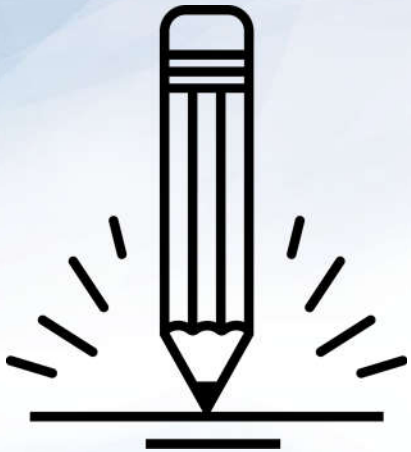
Instructor Demonstration

Functions

DRY, or Don't Repeat Yourself, is a software development principle that we use to reduce repetition in written code.



Get Ready for Some Live Coding!



Activity: Functions

In this activity, you will write a function that returns the arithmetic mean (average) for a list of numbers.

Suggested Time:
10 Minutes

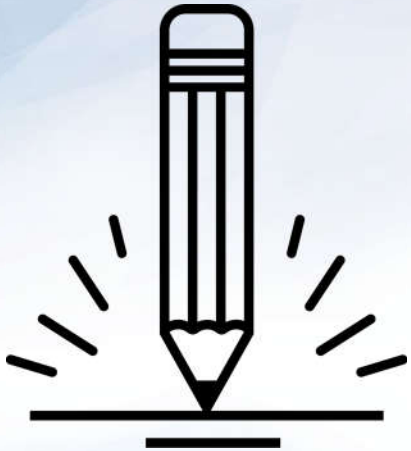
Instructions:

Activity: Functions

- Write a function called `average` that accepts a list of numbers as a single argument.
  - The function should return the arithmetic mean (average) for a list of numbers.

- Test your function by calling it with different values and printing the results.

Time's Up! Let's Review.



Activity: Graduating Functions

In this activity, you will create a function that searches a list of students and graduates by state to determine state graduation rates for public, private nonprofit, and private for-profit institutions.

Suggested Time:
15 Minutes

Instructions:

Activity: Graduating Functions

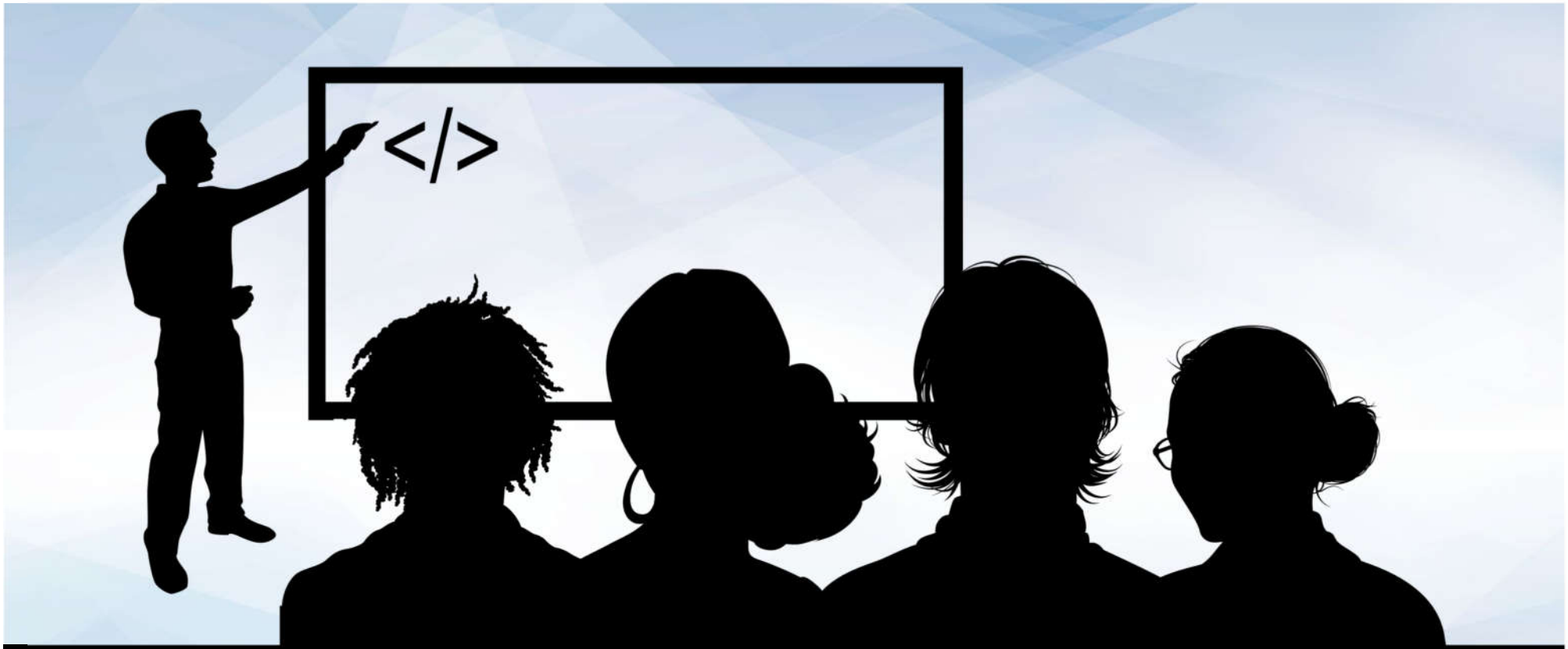
- Analyze the code and CSV provided to determine what needs to be added to the application.
- Using the starter code provided, create a function called `print_percentages` that takes in a parameter called `state_data` and does the following actions:
  - The function uses the data stored within `state_data` to calculate the estimated graduation rates in each category of Title IV 4-year institutions (public, nonprofit private, and for-profit private).
  - The function prints out the graduation rates for each school type for the state to the terminal.
- **Note:**
  - Some states do not have nonprofit or for-profit private schools, so data must be checked for zeros.
- **Bonus:**
  - Within the `print_percentages()` function, calculate the overall graduation rate. Then, create a conditional that checks a state's overall graduation rate and prints `"Graduation success"` if the number was higher than 50 or `"State needs improvement"` if the number was lower than 50.

Time's Up! Let's Review.

Countdown timer

40:00

(with alarm)



Instructor Demonstration

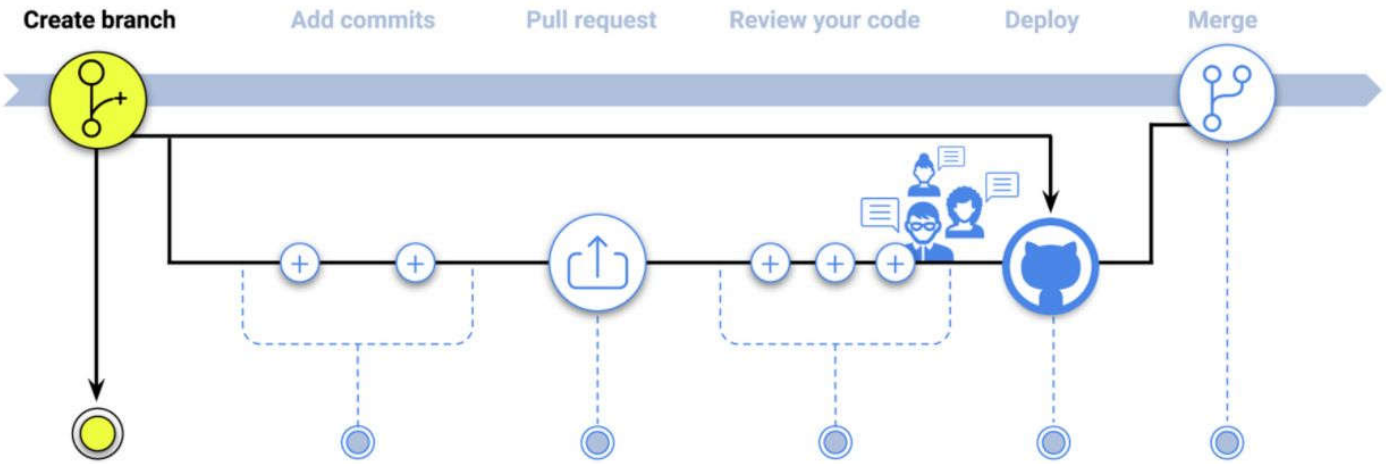
Intro to Git

Intro to Git

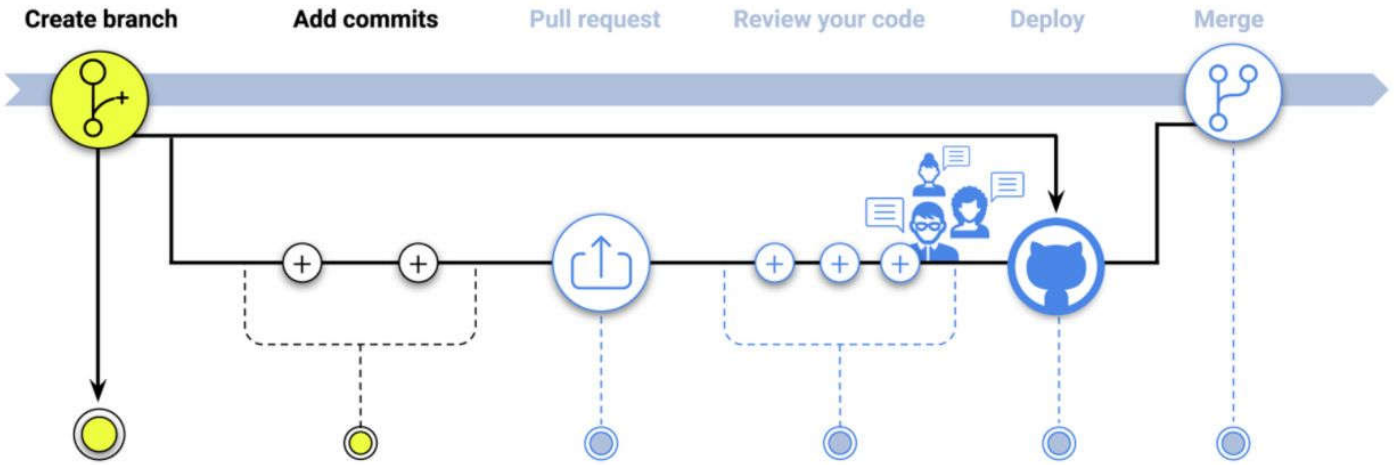
- More than 40 million people use GitHub. It's an essential tool for coding best practices, and the largest platform for developers to showcase their work.
- Using GitHub makes it easier to collaborate with colleagues and peers and to review previous versions of our work.

- GitHub is an essential tool for your academic development that you will take with you throughout your professional career.

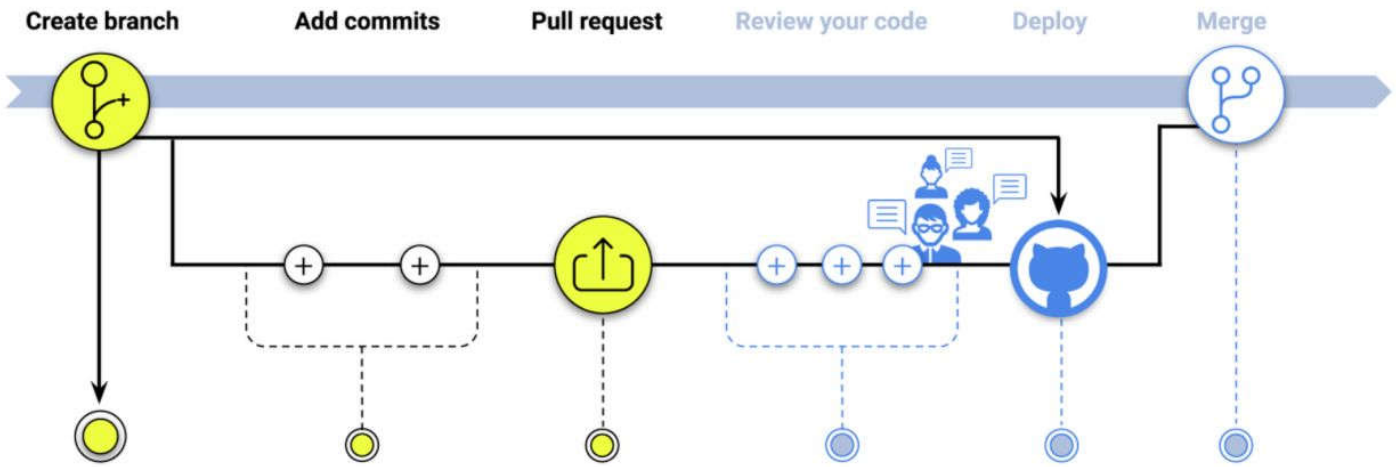
GitHub Flow: Create a Branch



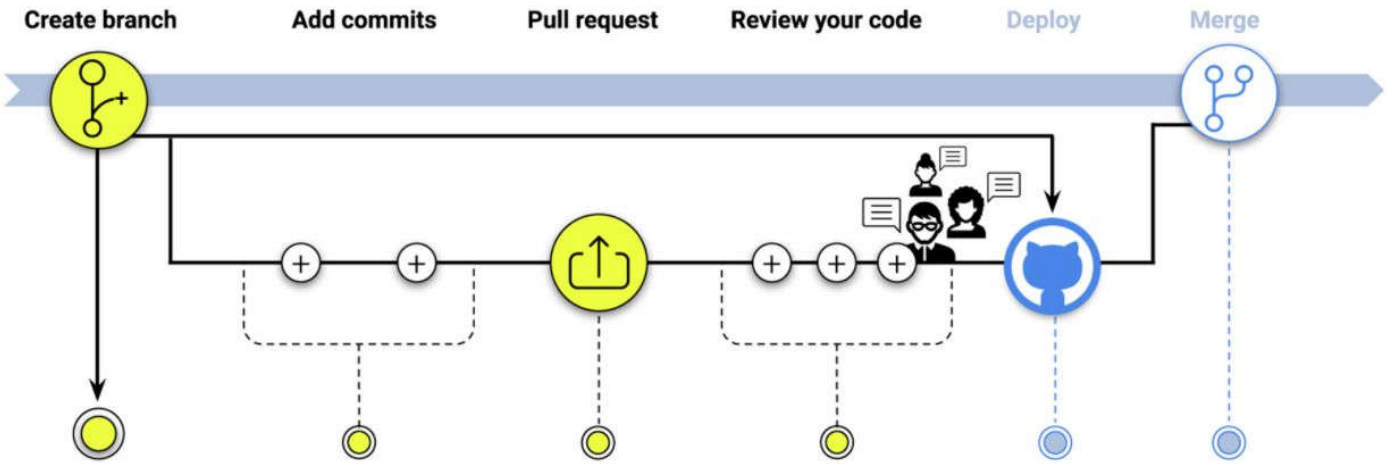
GitHub Flow: Add Commits



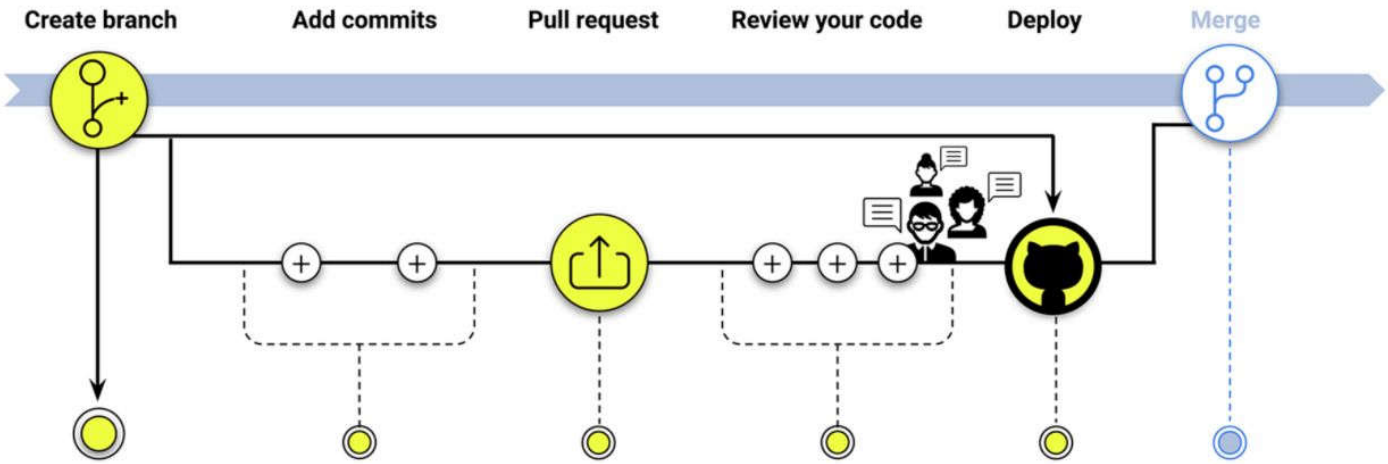
GitHub Flow: Pull Request



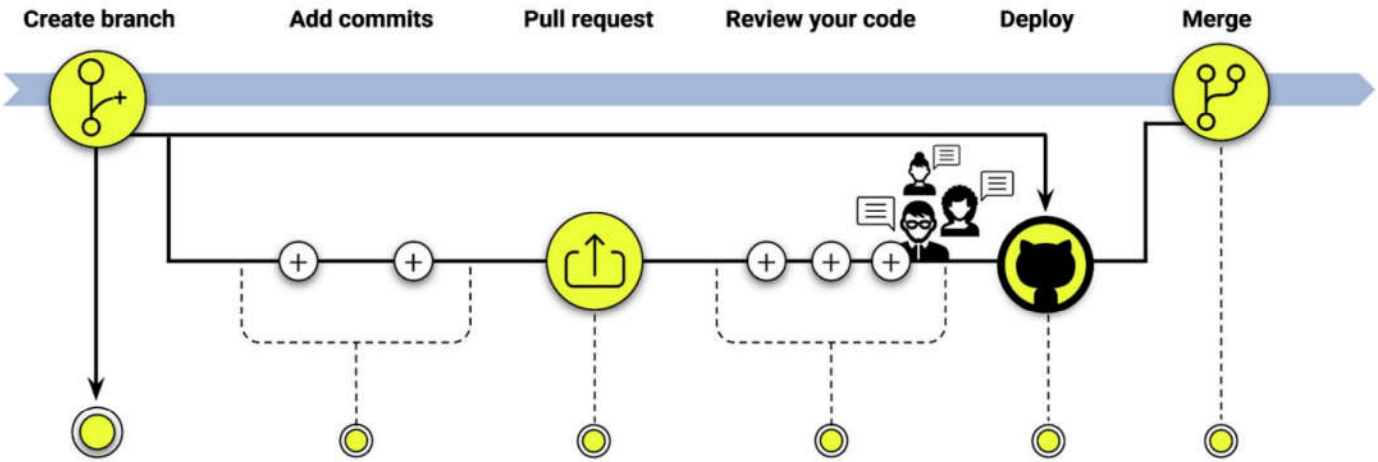
GitHub Flow: Review Your Code

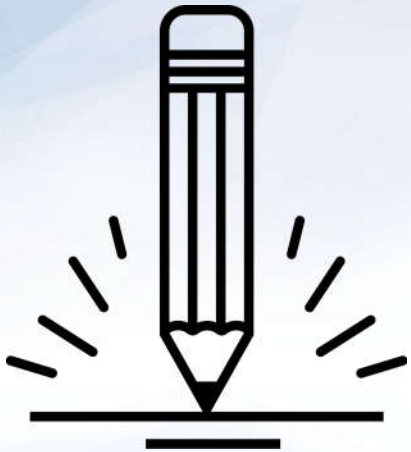


GitHub Flow: Deploy



GitHub Flow: Merge Your Code





Activity: Adding Files from the Command Line

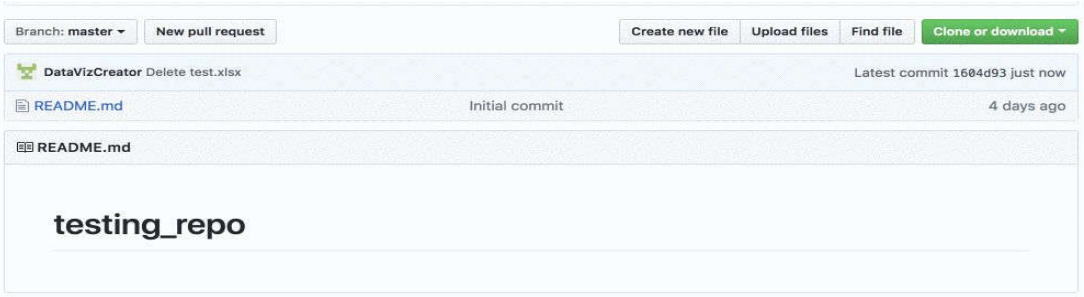
In this activity, we will create a new repository, clone it, and add files via the command line.

Suggested Time:
10 Minutes

Instructions

Activity: Adding Files from the Command Line

- Create a new repo.



- Open the terminal and navigate to the home folder by using `cd ~`.
- Type `git clone <repository link>` in the terminal to clone the repo to the current directory. Once this code has run, everyone should find a folder with the same name as the repo.

Instructions

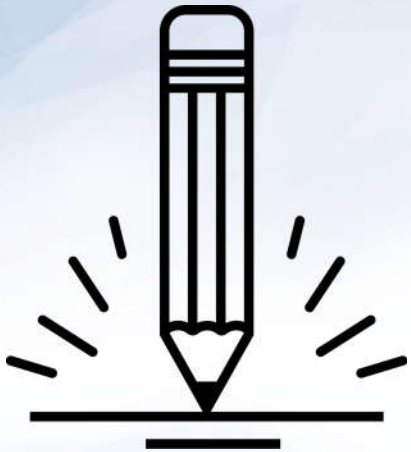
Activity: Adding Files from the Command Line

- Open the folder in VS Code, and create two Python script files named `script01.py` and `script02.py`.
- Once the files have been created, open the terminal and navigate to the repo folder. Run the following lines and explain each as you go through them.

```
# Displays that status of files in the folder
git status

# Adds all the files into a staging area
git add .

# Check that thr files were added correctly
git status

# Commits all the files to your repo and adds a message
git commit -m <add commit message here>

# Pushes the changes up to GitHub
git push origin master
```

Main Branch

- Historically, the most common name for the main body of a codebase has been **master**. Recently, however, **main** has become more popular.
- In fact, GitHub now uses **main** as the default name for its repositories, as do the projects in this course. You might encounter both terms throughout your development career, as experienced coders may use "master branch" out of habit.



Activity: Adding More to the Repo

In this activity, you will make or add changes to the repo that we just created.

Suggested Time:
15 Minutes

Instructions:

Activity: Adding More to the Repo

Using the repo that we just created, make or add the following changes:

- Add new lines of code to one of the Python files.
- Create a new folder.
- Add a file to the newly created folder.
- Add, commit, and push the changes.
- Delete the new folder.
- Add, commit, and push the changes again.

Time's Up! Let's Review.


The End