



# Python Deeper Dive

Data Boot Camp  
Lesson 3.3



# Class Objectives

---

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



Add, commit, and push code up to GitHub from the command line.



Create and use Python dictionaries.



Read data in from a dictionary.



Use list comprehensions.



Write and re use Python function.



Have a firm understanding of coding logic and reasoning.



## Activity: Cereal Cleaner

In this activity, you will be creating an application that reads in cereal data from CSV and then prints only those cereals that have more than 5 grams of fiber in them.

**Suggested Time:**  
20 Minutes



# Instructions: Activity: Cereal Cleaner

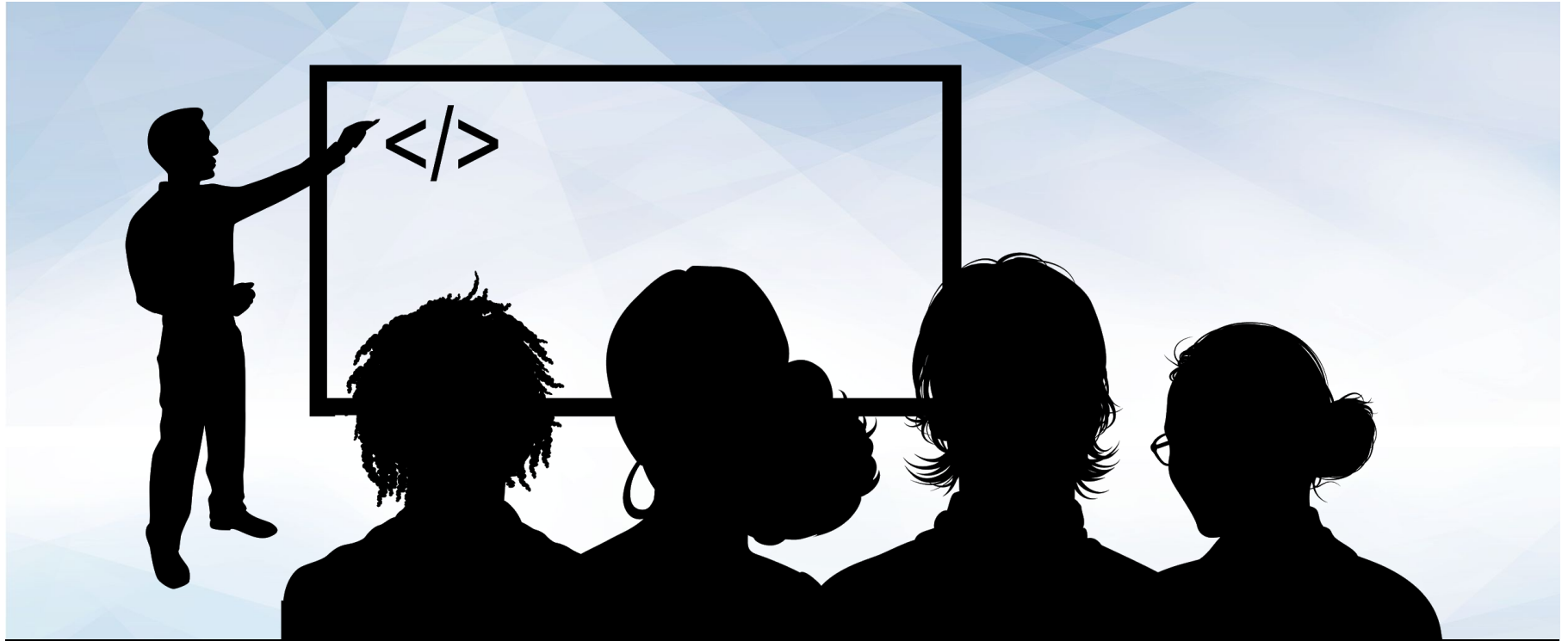
---

- Read through `cereal1.csv` and find the cereals that contain five grams of fiber or more, printing the data from those rows to the terminal.
- **Hint:**
  - 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 at the first row.
  - Note that `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 following again but this time using `cereal_bonus`, which does not include a header.





**Time's Up!** Let's Review.



# Instructor Demonstration

## Dictionaries



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



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

## Dictionaries

---

- Like lists and tuples, dictionaries can contain multiple values and data types within them.
- Unlike lists and tuples, however, dictionaries store data in key-value pairs. The key in a dictionary is a string that can be referenced in order to collect the value it is associated with.
- Very similar to a dictionary that contains definitions, 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 = {}
```

- Or, you can 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 value desired afterwards.
- Referencing a value within a dictionary is as simple as calling the dictionary and following it up with a pair of brackets containing the key for the value desired.

```
>>> {}
```

# 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, and 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 placing another key-value pair afterwards.
  - Keys are immutable objects, like integers, floating-point decimals, or strings. Keys cannot be lists or any other type of mutable object.
  - Values in a dictionary can be objects of any type: integers, floating-point decimals, strings, Boolean values, datetime values, and lists.

```
>>> # A list of actors  
>>> []
```

# Dictionaries

---

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

```
{
```

```
>>> {
```

```
>>> {
```



## Activity: Hobby-Book

In this activity, you will be creating and accessing your own dictionaries based upon your hobbies.

**Suggested Time:**  
10 Minutes



# Activity: Hobby-Book

---

- Create a dictionary that will store the following:
  - Your name
  - Your occupation
  - Your age
  - A list of a few of your hobbies
  - A dictionary of a few days and the time you wake up on those days
- Print out your name, your occupation, how many hobbies you have and a time you get 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`.
- Prompts the user for the names of five people they know and store them in a list.
- Run the provided program. Note that nothing forces you to write the name "properly"—e.g., 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-cased versions of each of the names in your lower-cased list.
- **Bonus:**
  - Instead of creating a lower-cased list and *then* a title-cased list, create the title-cased list in a single comprehension.
- **Hints:**
  - See the documentation for the `title` method.





**Time's Up!** Let's Review.

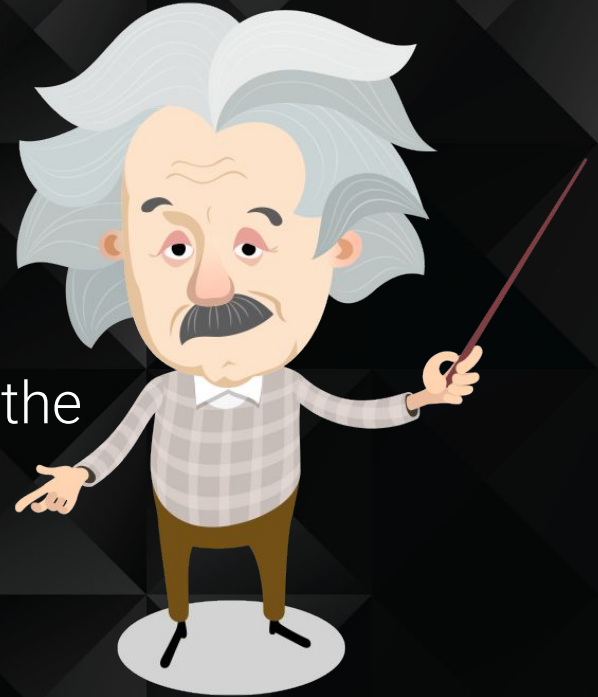


# Instructor Demonstration

## Functions



**DRY** - Don't Repeat Yourself is a principle of software development to reduce repetition of the written code.





Get ready  
for  
some live coding...





## Activity: Functions


In this activity, you will write a function that returns the arithmetic 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 `average` should return the arithmetic **mean** (average) for a list of numbers.
- Test your function by calling it with different values and printing the results.
- **Hint**
  - Arithmetic Mean (average) 



**Time's Up!** Let's Review.



## Activity: Wrestling With Functions

In this activity, you will create a function capable to search through a list of wrestlers to determine their win, loss, and draw percentages.

**Suggested Time:**  
15 Minutes



## Instructions:

# Activity: Wrestling With Functions

---

- Analyze the code and CSV provided, looking specifically for what needs to still be added to the application.
  - Note that `header = next(reader)` will read the header row from the csv file.
- Using the starter code provided, create a function called `print_percentages` which takes in a parameter called `wrestler_data` and does the following:
  - Uses the data stored within `wrestler_data` to calculate the percentage of matches the wrestler won, lost, and drew over the course of a year.
  - Prints out the stats for the wrestler to the terminal
- **Bonus:**
  - Still within the `print_percentages()` function, create a conditional that checks a wrestler's loss percentage and prints either "Jobber" to the screen if the number was greater than fifty or "Superstar" if the number was less than 50.



**Time's Up!** Let's Review.



Countdown timer

**40:00**

(with alarm)





# Instructor Demonstration

## Intro to Git



## Activity: Adding Files from the Command Line

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

**Suggested Time:**  
10 Minutes

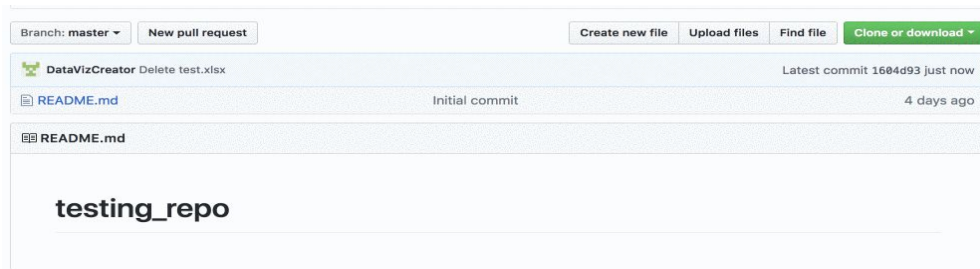


# Instructions:

## Activity: Adding Files from the Command Line

---

- Create a new repo.



- Open terminal (or git-bash for Windows users) and navigate to the home folder using `cd ~` (or another directory where you want to store your code).
- Type in `git clone <repository link>` in the terminal to clone the repo to the current directory. Once this has run, everyone should now see a folder with the same name as the repo.

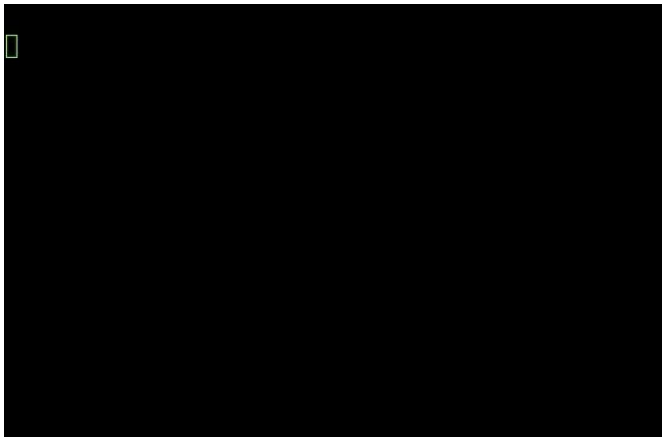


# Instructions:

## 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 up Terminal/git-bash 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
```

## Activity: Adding Files from the Command Line

---

- Historically, the most common name for the main body of a codebase has been **master**. Recently, however, **main** has been gaining in popularity. In fact, GitHub now uses **main** as the default name for its repositories - as do the projects in this course. Be aware that you might see instances of both throughout your development career, or hear experienced coders use the term "master branch" out of habit.



## Activity: Adding more to the repo.

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

**Suggested Time:**  
10 Minutes



# Instructions:

## Activity: Adding more to the repo

---

- Using the repo that 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.



# What's GitHub? Intro to Git

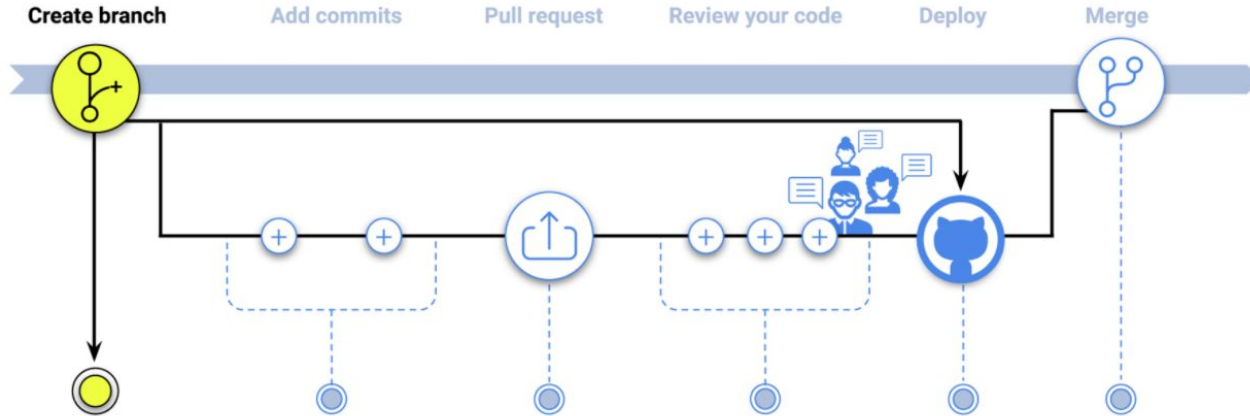
---



- More than 40 million people use GitHub.
  - It is an essential tool for best code practices.
  - It is the biggest platform for developers to showcase their work.
  - Using GitHub makes it easier to collaborate with colleagues and peers and to review previous versions of one's work.
- 
- GitHub is an essential tool for your academic development, and it will transition with you through your professional career.
  - GitHub can raise your visibility among potential employers and could translate into a high-paying job.

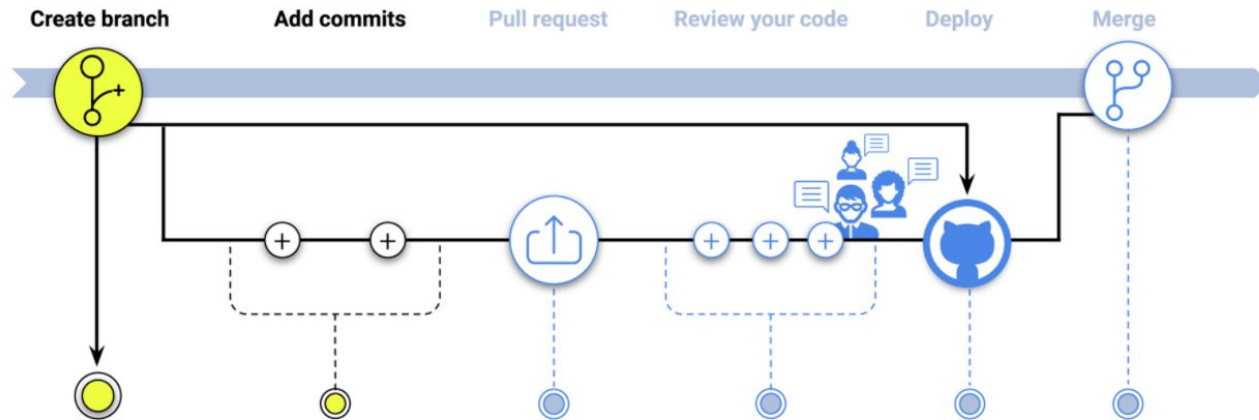
# GitHub Flow: Create a Branch

## Intro to Git



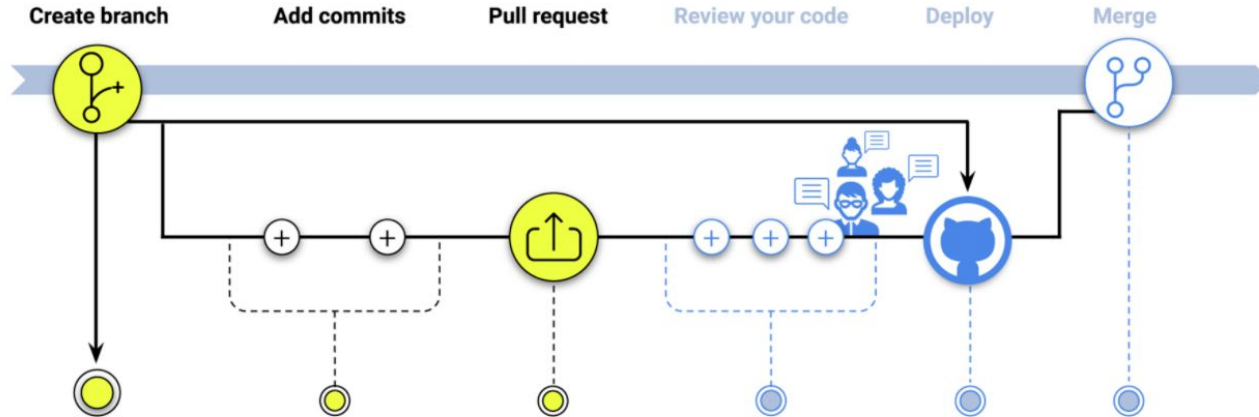
# GitHub Flow: Add Commits

## Intro to Git



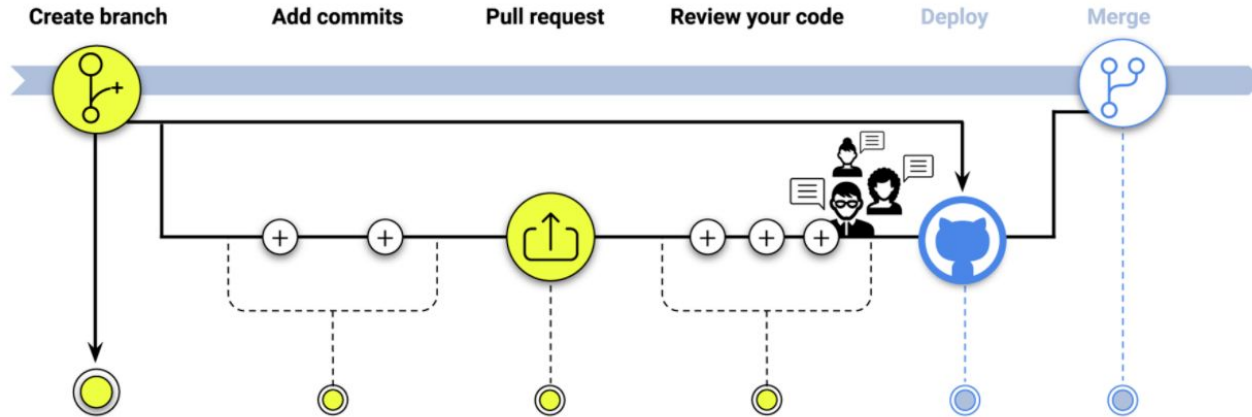
# GitHub Flow: Pull Request

## Intro to Git



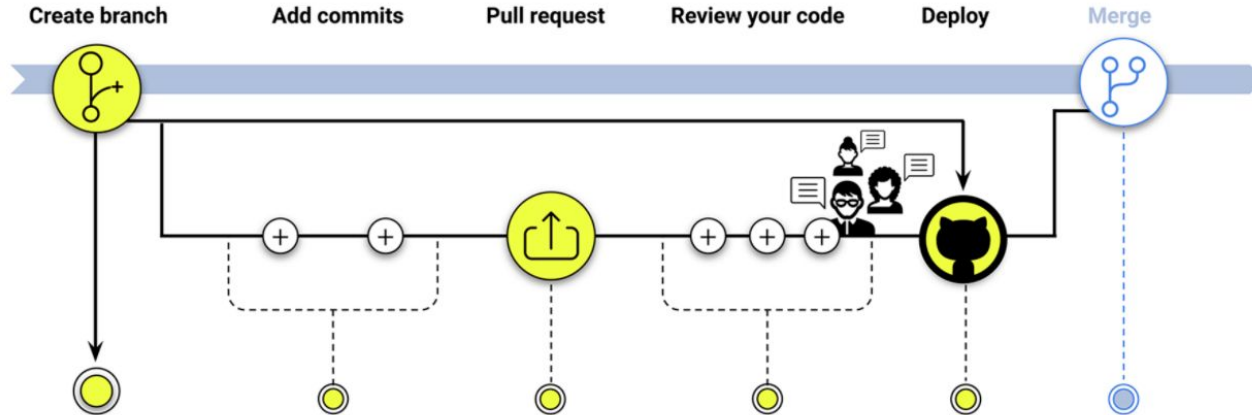
# GitHub Flow: Review your Code

## Intro to Git



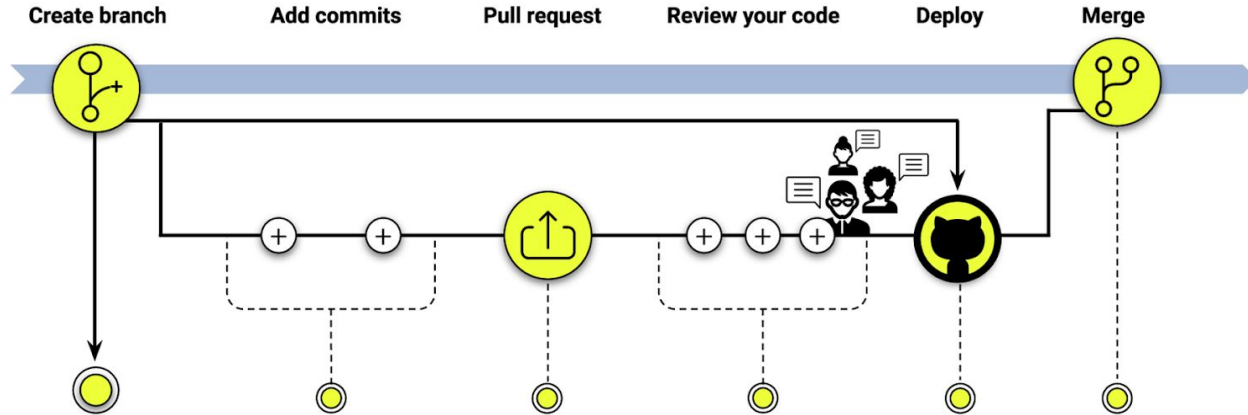
# GitHub Flow: Deploy

## Intro to Git



# GitHub Flow: Merge Your Code

## Intro to Git



last  
but  
not least..



>>>

>>>

