Introduction to Flask & Serving Data with APIs

Data Boot Camp

Lesson 10.3

Class Objectives

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

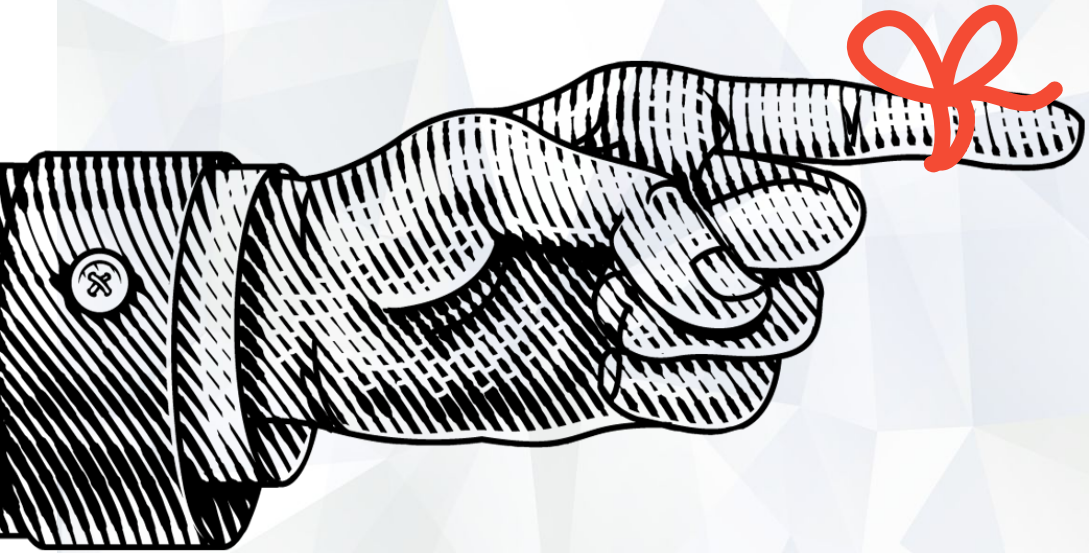
- Use Flask to create and run a server.
- Define endpoints using Flask's `@app.route` decorator.
- Extract query-variable path values from get requests.
- Use variable paths to execute database queries on behalf of the client.
- Return JSONified query results from API endpoints.

WELCOME

Joins & Dates

We will learn how to perform joins in

SQLAlchemy



Remember,

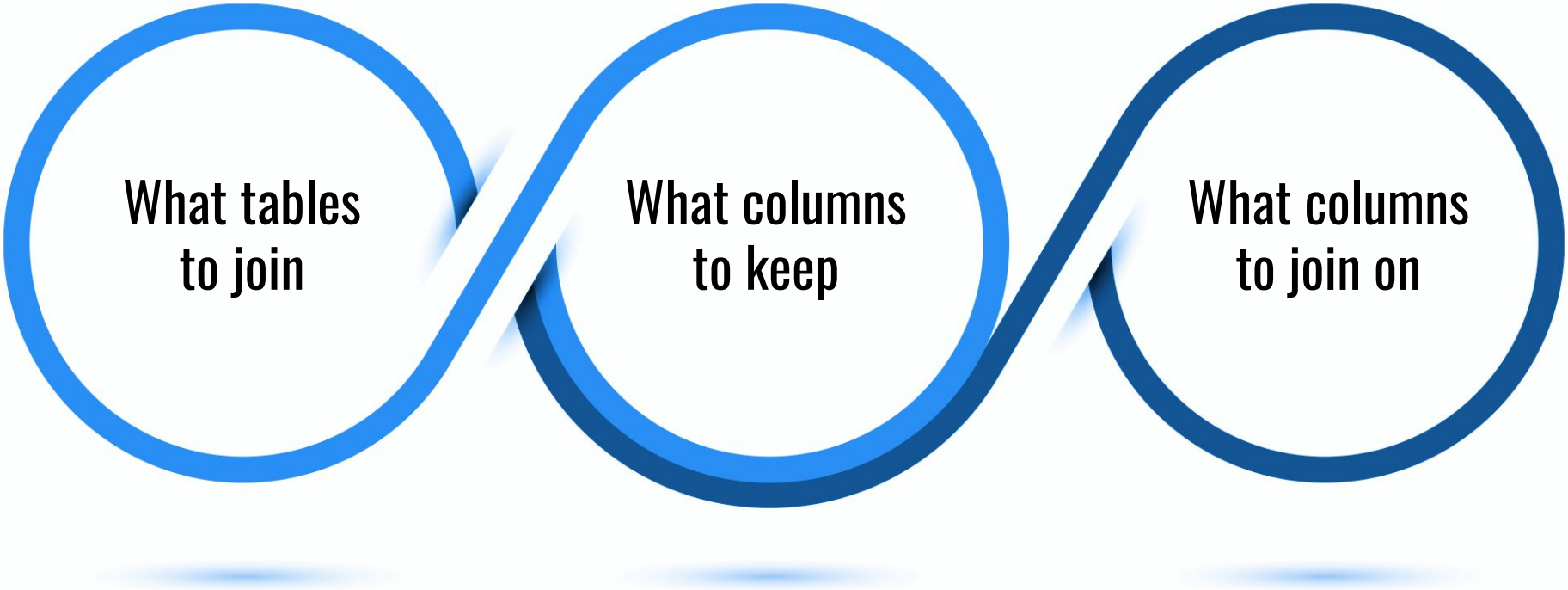
SQLAlchemy can use pure SQL to manipulate SQL databases, or a more Pythonic object-based approach can be used.

When we use Python classes and objects with SQLAlchemy, SQL query joins behave similarly to Pandas DataFrame joins.

To perform a join on tables using SQLAlchemy, we must first identify the structure and data contained within the SQL tables.

Joins & Dates

We can use the `.filter() method` to obtain the merged table results once we identify:



Instructor Demonstration

Joins

Questions?

Instructor Demonstration

Dates

Questions?

Activity: Dates

In this activity, you will practice working with dates, both in SQLAlchemy and with the datetime library.

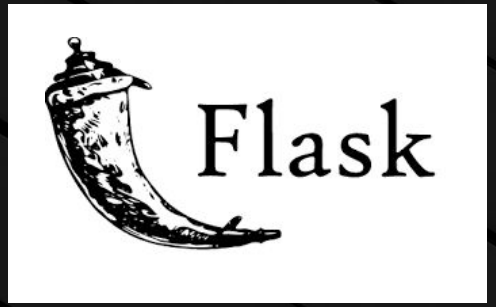
Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

Questions?

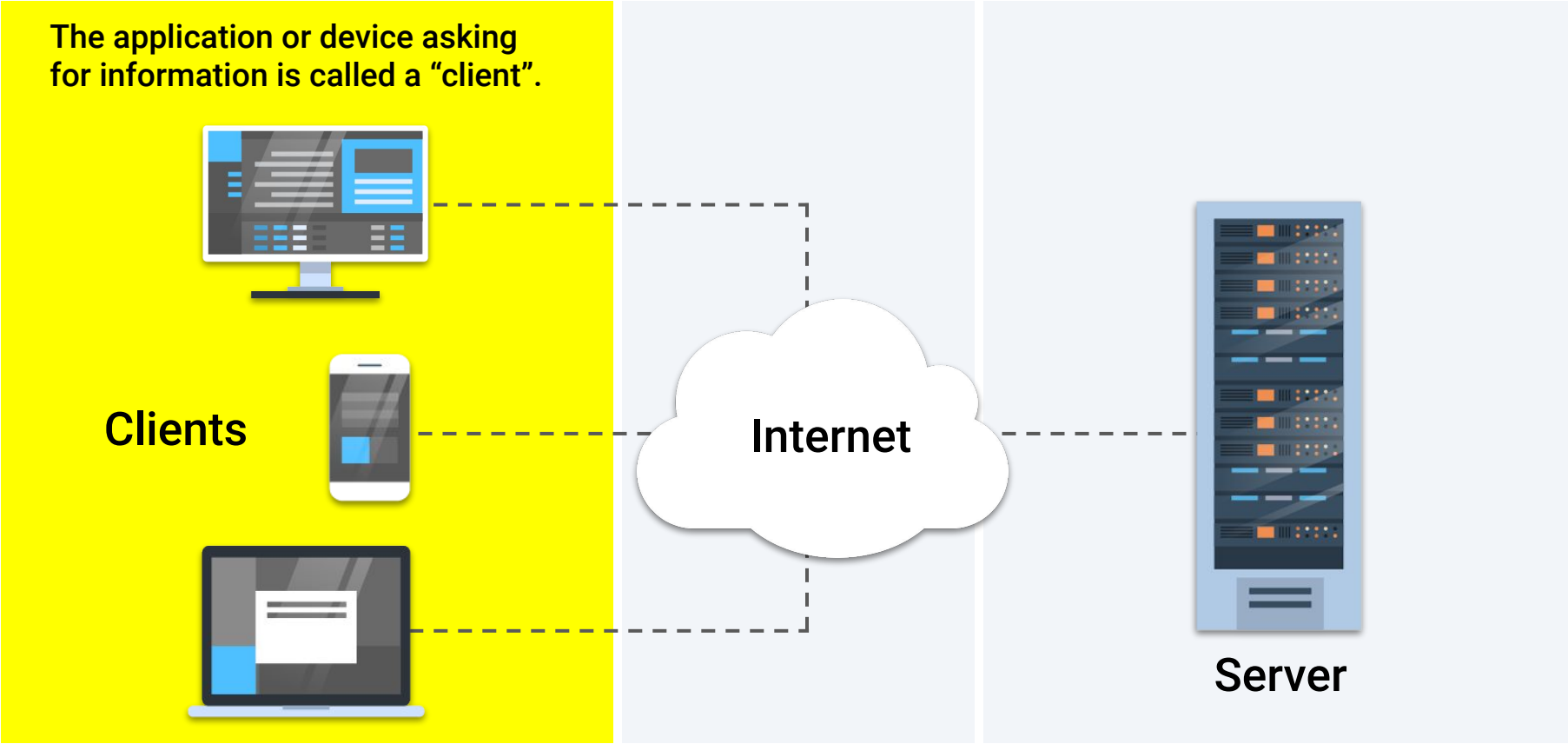
Introduction to



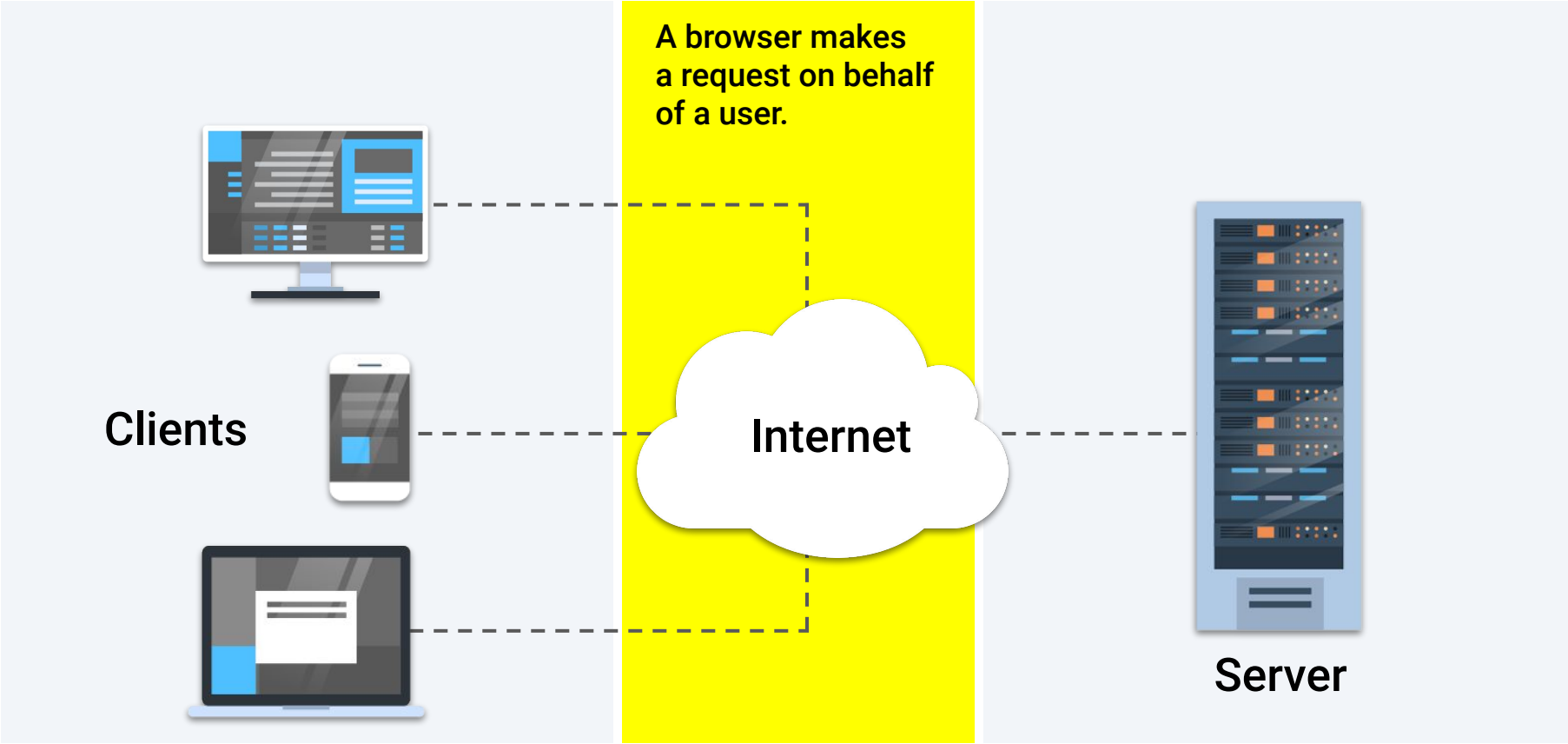
The internet is built on a model of clients requesting data from servers.

(A server is, essentially, just a program.)

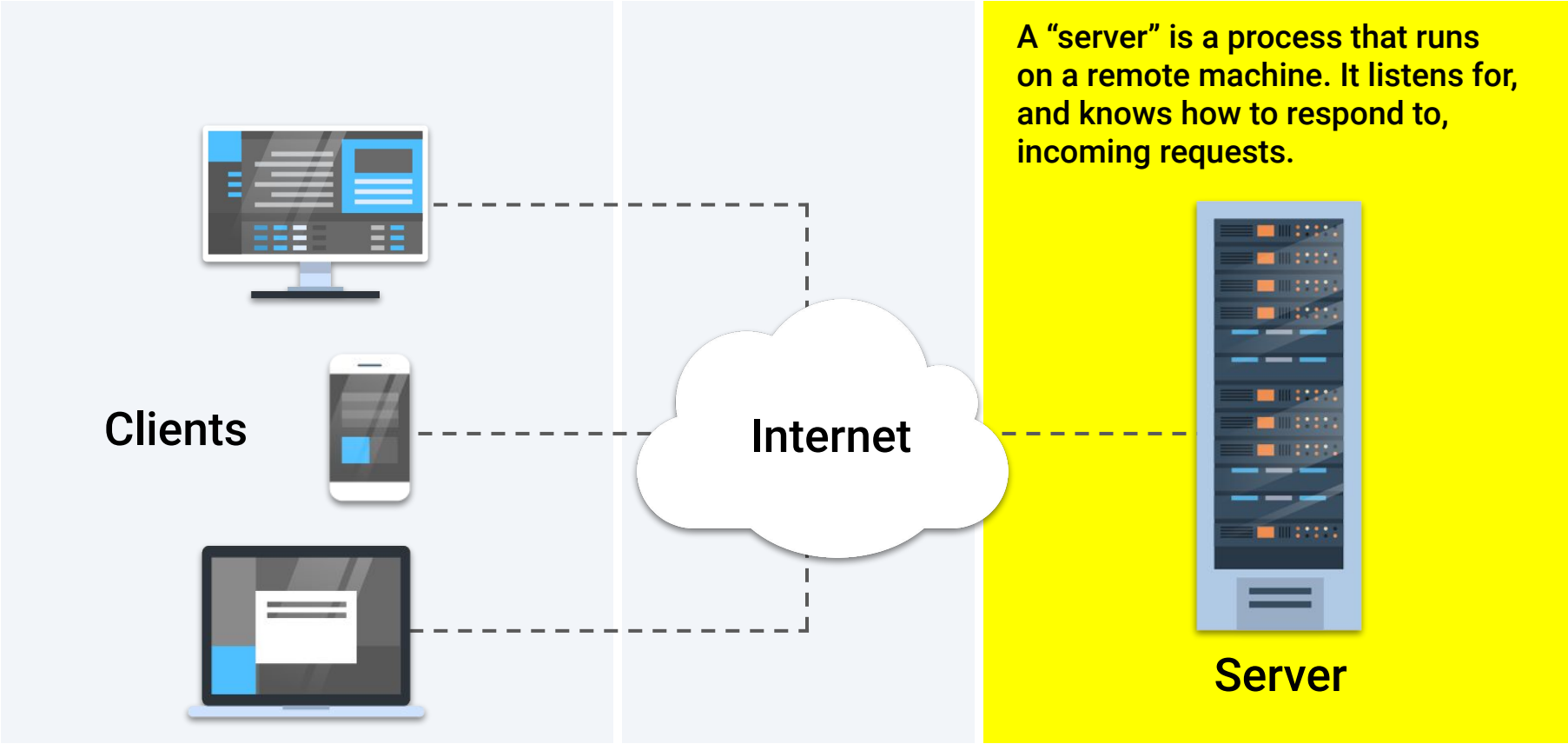
The Internet is Built from Clients and Servers



The Internet is Built from Clients and Servers



The Internet is Built from Clients and Servers



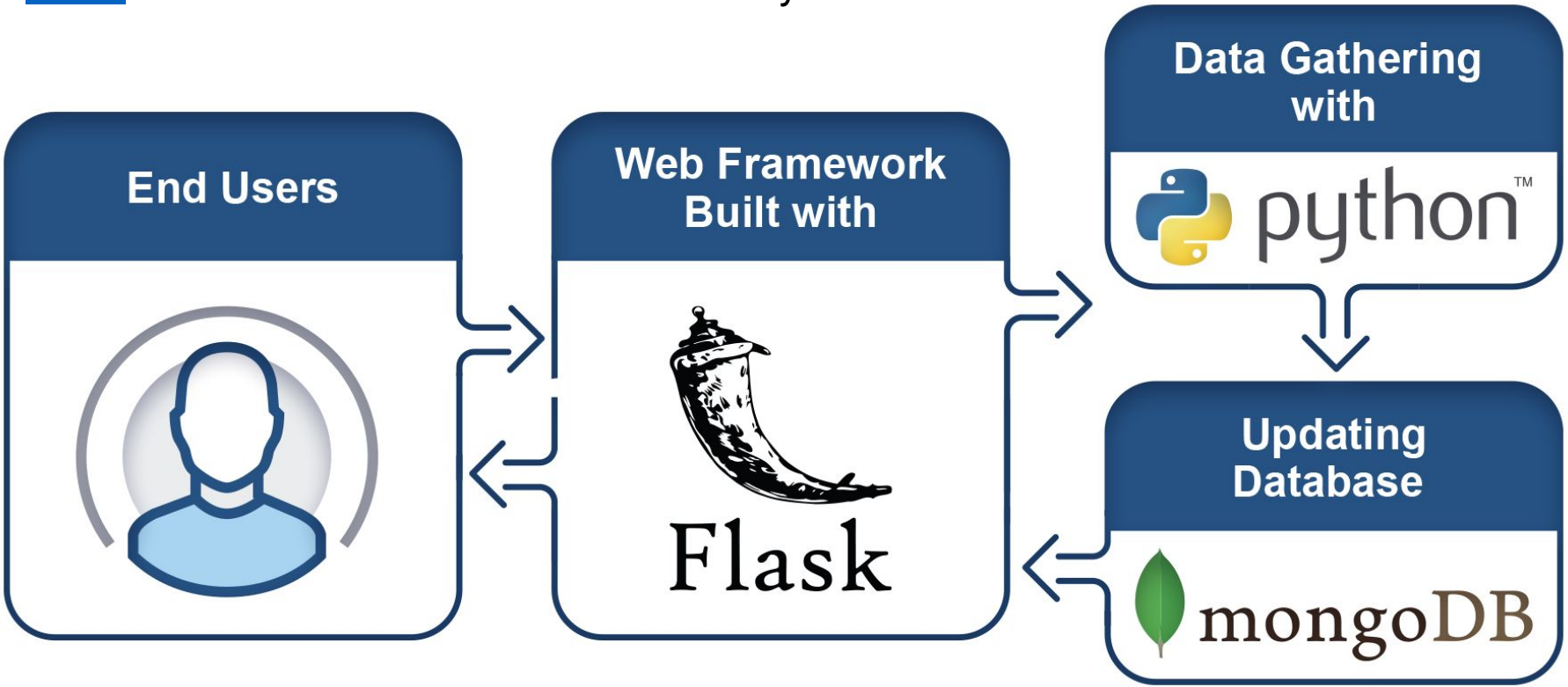
Introduction to Flask

When we create an API for others to use, the code they write acts as a client to our API server.

- We have no control over the code that our consumers write.
- As API developers, we do not write client code. We will focus on writing the code that runs the server.
- This is the code responsible for retrieving and returning the data requested by users.

Introduction to Flask

Flask is a micro web framework to build your own APIs!



Instructor Demonstration

Introduction to Flask

Questions?

Activity: Hello, Web

In this activity, you will practice setting up a server and defining basic routes with Flask.

Suggested Time:

10 Minutes

Time’s Up! Let’s Review.

Justice League

JSON APIs with jsonify

- All of the routes that we've written so far have returned *string* responses.
- The APIs we've dealt with *do not* return raw text; rather, they return JSON data.
- Fortunately, Python dictionaries map naturally to JSON.

Flask has a built-in method to automatically convert a dictionary into a properly formatted JSON response: `jsonify`.

Flask Has a Function to Create JSON Responses

We can use jsonify to create an HTTP response with the dictionary data that we want to send back to the client.

```
from flask import Flask, jsonify


app = Flask(_name_)

hello_list = ["Hello", "World"]

@app.route("/")
def home():
    return "Hi"

@app.route("/normal")
def normal():
    return str(hello_list)

@app.route("/jsonified")
def jsonified_list():
    return jsonify(hello_list)
```

Flask Has a Function to Create JSON Responses

`jsonify` automatically converts Python lists into JSON responses.

```
from flask import Flask, jsonify


app = Flask(_name_)

hello_list = ["Hello", "World!"]

@app.route("/")
def home():
    return "Hi"

@app.route("/normal")
def normal():
    return str(hello_list)

@app.route("/jsonified")
def jsonified_list():
    return jsonify(hello_list)
```

Flask has a function to create JSON responses

When converting Python dictionaries into JSON responses, `jsonify` is not needed.

```
from flask import Flask, jsonify


app = Flask(_name_)

hello_dict = {"Hello": "World!"}

@app.route("/")
def home():
    return "Hi"

@app.route("/normal")
def normal():
    return str(hello_dict)

@app.route("/dict")
def dictionary():
    return hello_dict
```

The converted JSON responses are wrapped in HTTP to send back to the client.

Instructor Demonstration

JSON APIs with jsonify

Questions?

Activity: Justice League

In this activity, you will create an API route that returns the superhero name and real name for every member of the Justice League.

Suggested Time:

20 Minutes

Time’s Up! Let’s Review.

Questions?


Break

Routes with Variable Rules

Our current API is one-dimensional

Our current API can only return the ***entire*** Justice League dataset.
It would be better if users could specify a particular character of interest.

```
@app.route("/api/v1.0/justice-league")
def justice_league():
      """Return the justice league data as json"""

   return jsonify(justice_league_members)
```

Our current API is one-dimensional

Ideally, consumers would be able to specify a character of interest in the URL, they could expect either of the following outcomes:

01 A JSON response with the character data if it's in the dataset; or

02 A JSON response with error information, indicating that the server couldn't find the character that the user requested.

```
return {"error": f"Character with real_name
{real_name} not found."}), 404
```

Instructor Demonstration

Routes with Variable Paths

Activity: Routes with Variable Rules

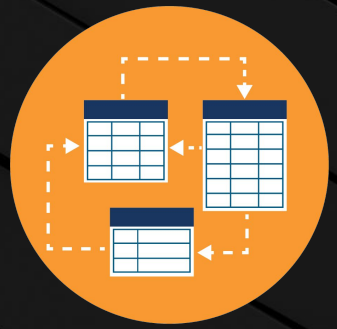
In this activity, you will add an additional API route that returns a JSON containing an individual superheroes information.

Suggested Time:

20 minutes

Time’s Up! Let’s Review.

Questions?



Chinook Database Analysis

Instructor Demonstration

Flask with ORM

Questions?