
INSTALLING PYTHON



CHECK IF YOU ALREADY HAVE PYTHON 3 INSTALLED

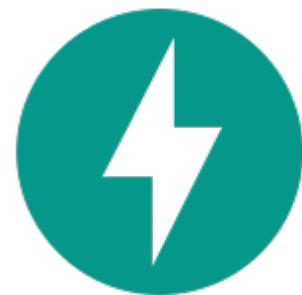
- Inside your “terminal” for Mac, or “command prompt” for windows.
- Type “python --version”



A screenshot of a terminal window with a black background and three colored dots (red, yellow, green) at the top. The text "python --version" is typed in blue, and the output "Python 3.9.1" is displayed in red.

```
python --version
Python 3.9.1
```

PYTHON INTEGRATED DEVELOPMENT ENVIRONMENT



FastAPI

WHAT ARE IDES?

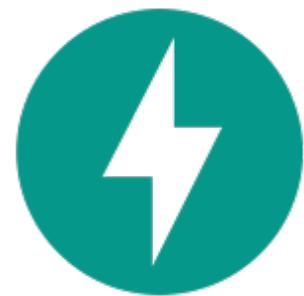
- IDEs are simply a source code editor. This means an IDE will help and assist in writing software!
- Many of them have terminals and other useful build automation tools embedded within.
- IDEs have debugger tools that allow you to dive deep within the code to find bugs and create solutions

WHY DO DEVELOPERS USE IDES?

- IDEs allow developers to begin work quickly and efficiently due to all the tools that come out of the box
- IDEs also parse all code, and helps find bugs before they are committed by human error
- The GUIs of IDEs is created with one goal in mind, help developers be efficient.



SETTING UP A PYTHON IDE



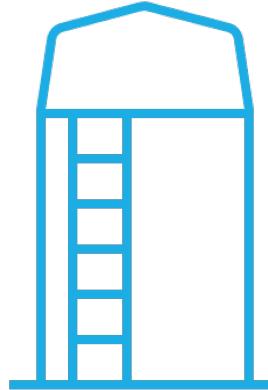
FastAPI

PYTHON VIRTUAL ENVIRONMENTS

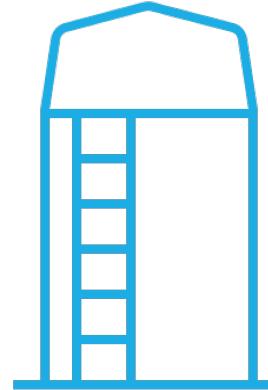


PYTHON VIRTUAL ENVIRONMENT

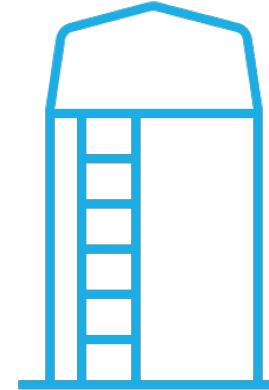
- A virtual environment is a Python environment that is isolated from those in other Python environments.



FastAPI



A.I.



Internet of Things

HOW ARE WE GOING TO INSTALL THE DEPENDENCIES?

- Pip: Pip is the Python package manager.
- Pip is used to install and update packages.
- You will want to make sure you have the latest version of pip installed

Unix/Mac OS



```
python3 -m pip --version  
# pip 21.2.4
```

Windows

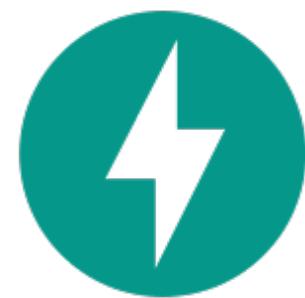


```
python -m pip --version  
# pip 21.2.4
```

HOW WILL WE BE SETTING UP THE VIRTUAL ENVIRONMENT

- I started by creating a brand-new folder / directory called “FastAPI”
- Within our pip we will be checking what dependencies we already have installed.
- Installing Virtual Env if it is not installed.
- Creating a new FastAPI environment as a virtual environment
- Activating our virtual environment
- Lastly, installing FastAPI to our virtual environment

SWAGGER, OPENAPI, REQUEST METHODS & STATUS CODES



FastAPI

OPENAPI SPECIFICATION (OAS):

- OpenAPI Document Defines:
 - Schema
 - Data Format
 - Data Type
 - Path
 - Object
 - And much more



VIEW OPENAPI SCHEMA

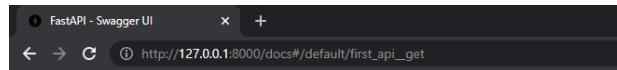
- FastAPI generates the OpenAPI schema so you can view
- <http://127.0.0.1:8000/openapi.json>
- Helps the developer create RESTful APIs based on standards so individuals can use the APIs easily



A screenshot of a terminal window on a Mac OS X system, indicated by the red, yellow, and green window control buttons at the top. The terminal displays a portion of an OpenAPI JSON schema. The schema starts with the root object and includes definitions for the API title, version, paths, and a specific endpoint's get method with its summary, operation ID, responses (including a 200 status code), and content type.

```
{  
  "openapi": "3.0.2",  
  "info": {  
    "title": "FastAPI",  
    "version": "0.1.0"  
  },  
  "paths": {  
    "/": {  
      "get": {  
        "summary": "First Api",  
        "operationId": "first_api__get",  
        "responses": {  
          "200": {  
            "description": "Successful Response",  
            "content": {  
              "application/json": {  
                "schema": {}  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

SWAGGER-UI



http://127.0.0.1:8000/docs

FastAPI 0.1.0 OAS3
[/openapi.json](#)

default

GET / First Api

Parameters

No parameters

Responses

Code	Description	Links
200	Successful Response	No links

Media type

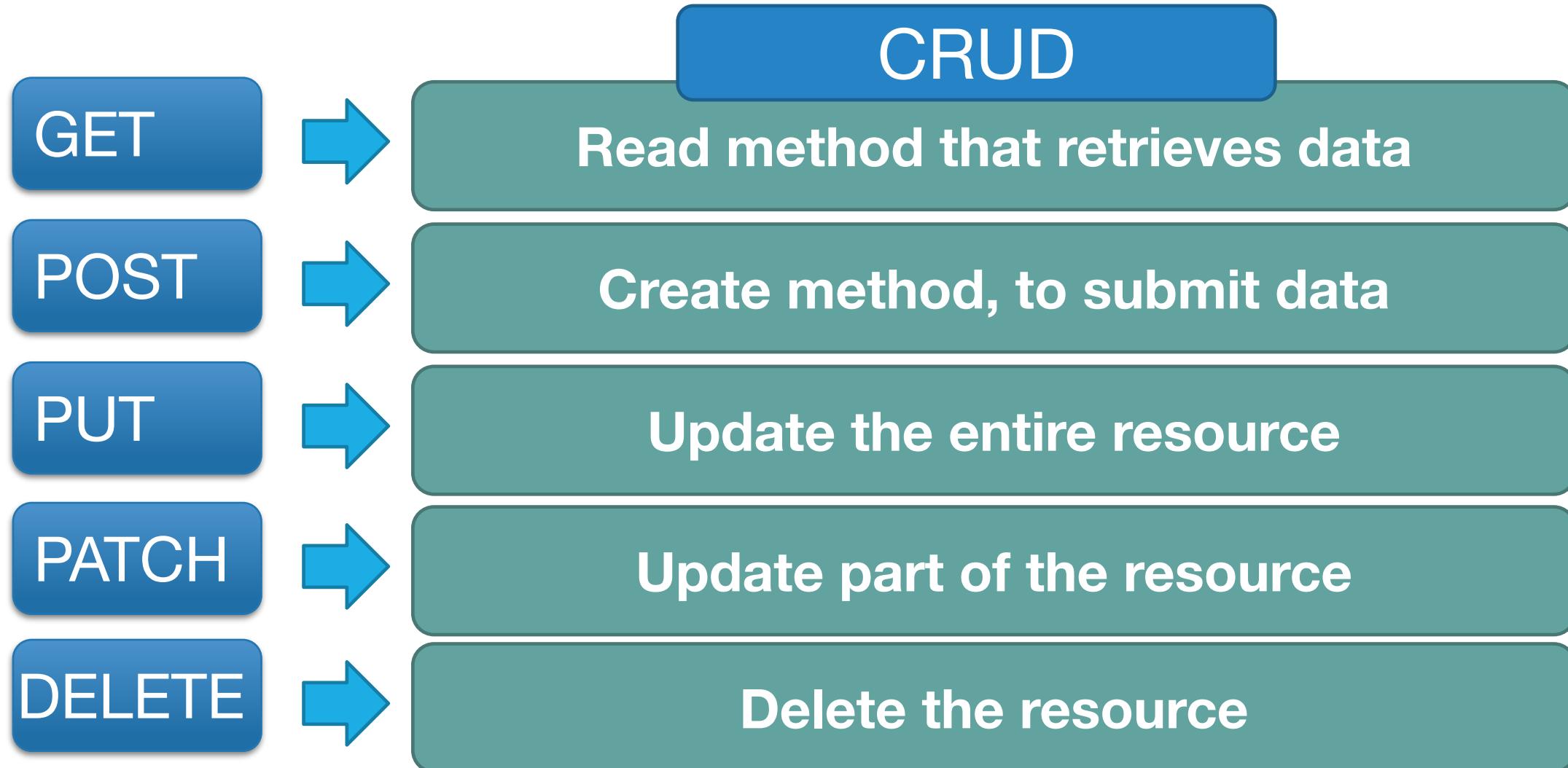
application/json

Controls Accept header.

Example Value | Schema

"string"

FASTAPI USES HTTP REQUEST METHODS



FASTAPI REQUEST METHODS (CONT)

TRACE



Performs a message loop-back to the target

OPTIONS



Describes communication options to the target

CONNECT



Creates a tunnel to the server, based on the target resource

GET



CRUD

Read method that retrieves data

POST



Create method, to submit data

PUT



Update the entire resource

PATCH



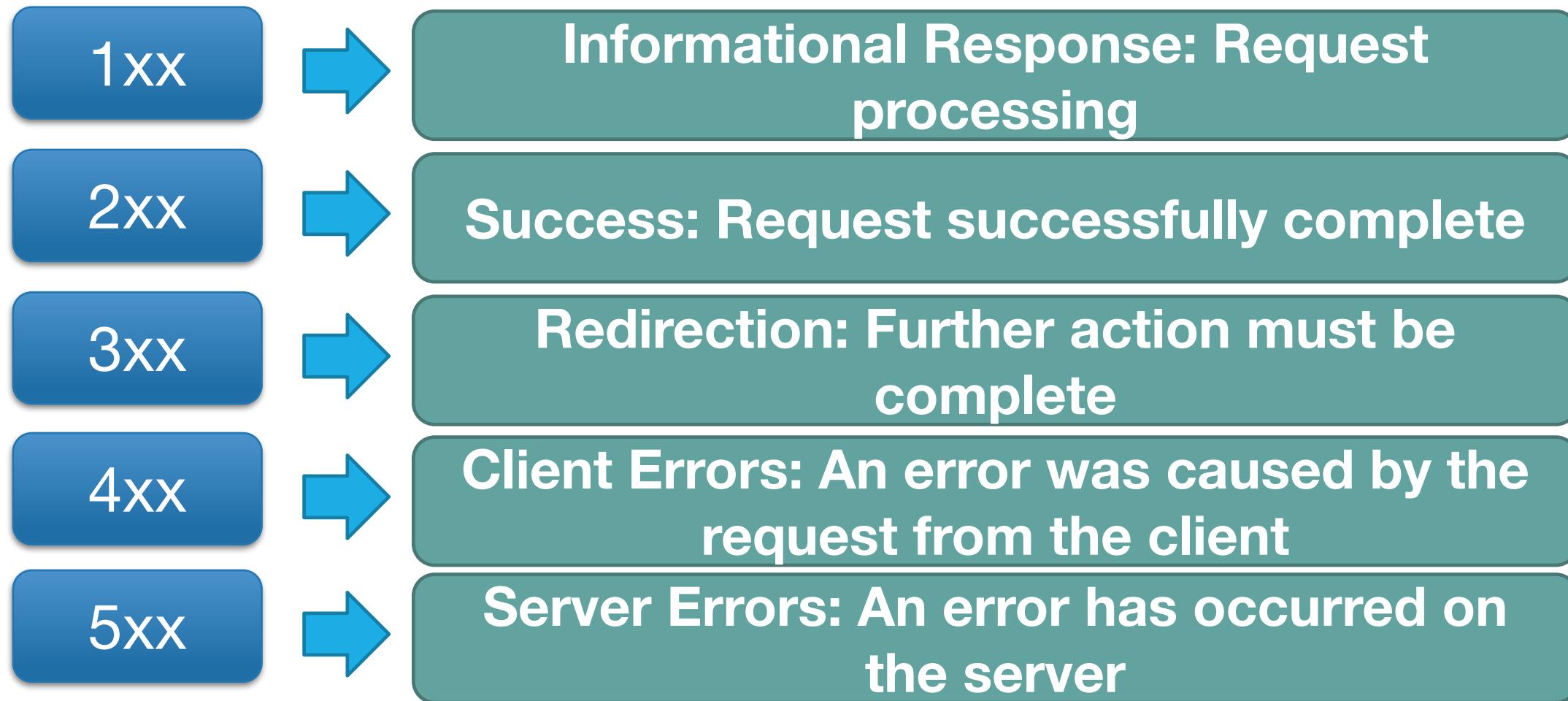
Update part of the resource

DELETE

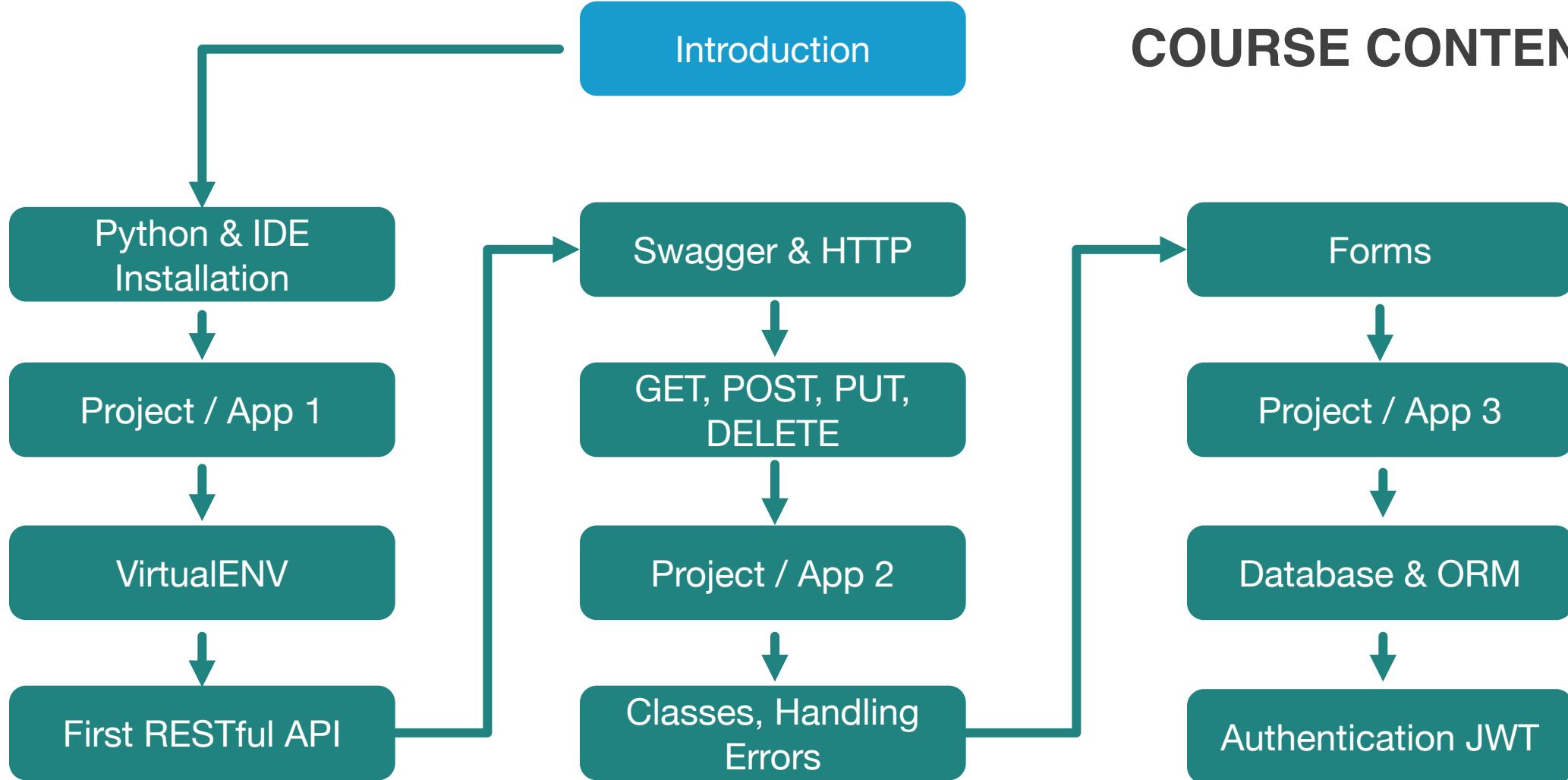


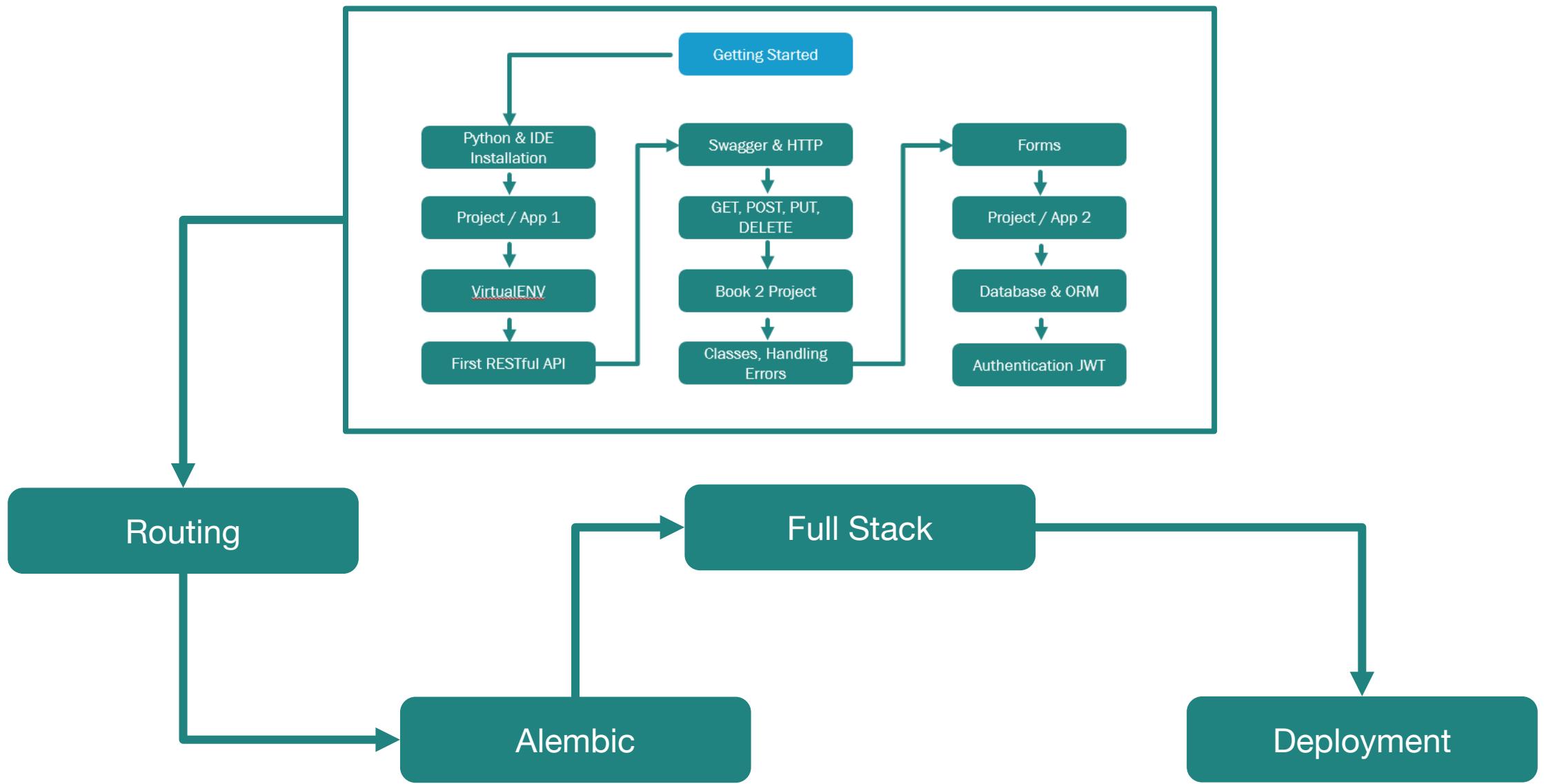
Delete the resource

FASTAPI RESPONSE STATUS CODE

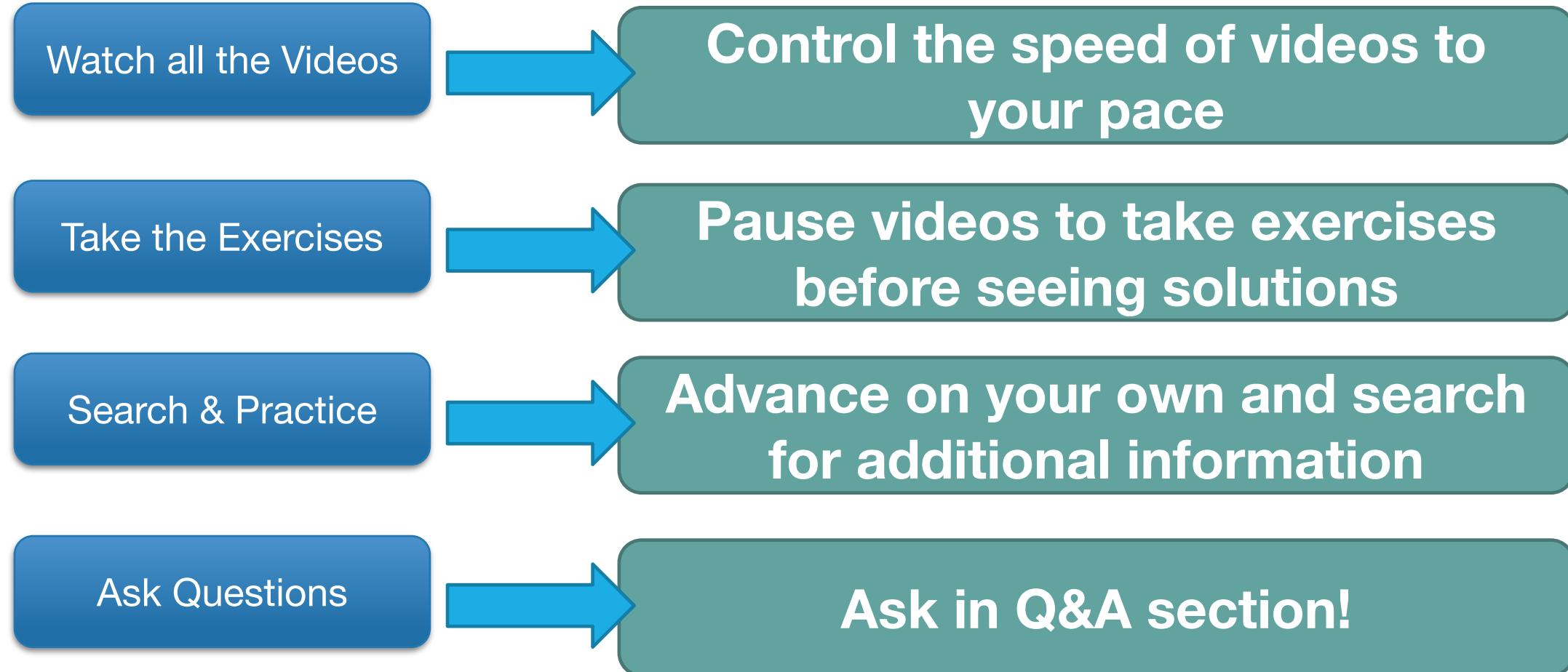


COURSE CONTENT!





HOW TO GET THE MOST OUT OF THE COURSE!

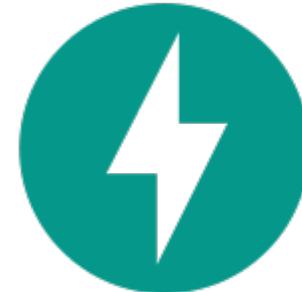


FASTAPI OVERVIEW



WHAT IS FASTAPI?

- FastAPI is a Python web-framework for building modern APIs
 - Fast (Performance)
 - Fast (Development)
- Key Notes:
 - Few Bugs
 - Quick & Easy
 - Robust
 - Standards



FastAPI

WHAT DOES THIS ALL MEAN FOR YOU?

- FastAPI is a web-framework for building modern RESTful APIs



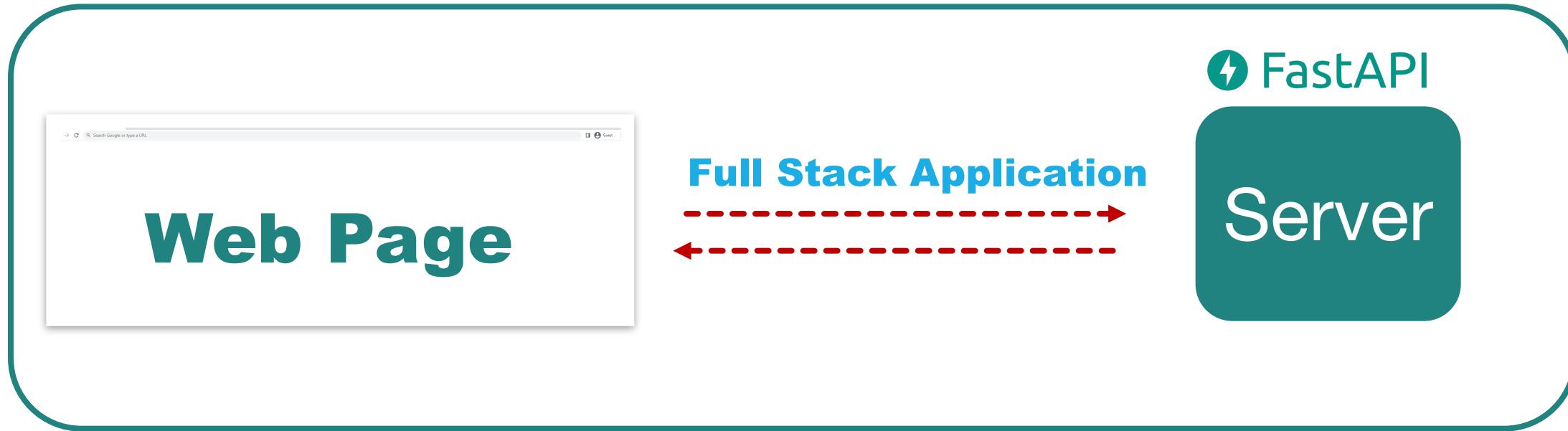
FastAPI

<https://fastapi.tiangolo.com/>



Official
documentation

WHERE DOES FASTAPI FIT WITHIN AN APPLICATION ARCHITECTURE?

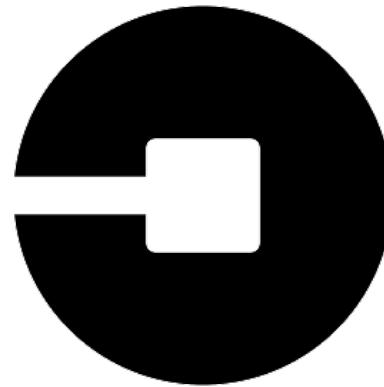


Handles all business logic for the application

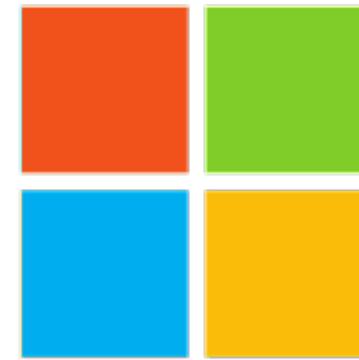
WHO HAS USED FASTAPI?



Netflix



Uber



Microsoft

CAN'T I JUST WRITE EVERYTHING MYSELF?

- FAQ: Why do I need a web framework?
 - You may be able to write everything yourself, but why reinvent the wheel?
 - Web-frameworks allow a simplified way for rapid development.
 - Includes many years of development, which allows you to have a secure and fast application! ☺

Books Project



What will we be creating?

- Creating and enhancing books to learn the basics of FastAPI

```
BOOKS = {  
    {'title': 'Title One', 'author': 'Author One', 'category': 'science'},  
    {'title': 'Title Two', 'author': 'Author Two', 'category': 'science'},  
    {'title': 'Title Three', 'author': 'Author Three', 'category':  
        'history'},  
    {'title': 'Title Four', 'author': 'Author Four', 'category': 'math'},  
    {'title': 'Title Five', 'author': 'Author Five', 'category': 'math'},
```

CRUD Operations

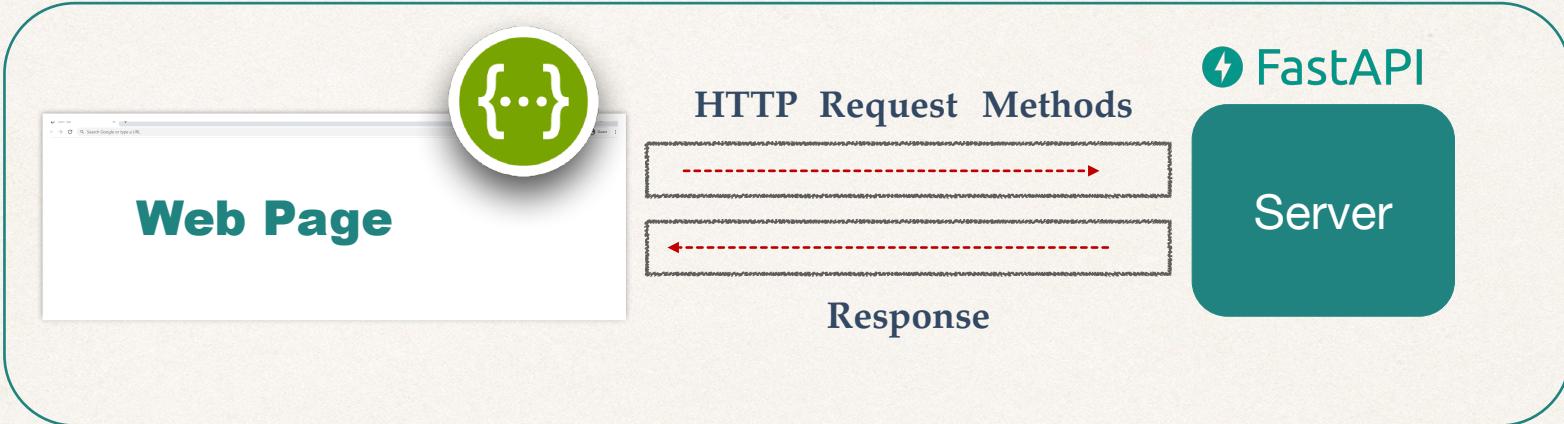
Create

Read

Update

Delete

Request and Response



CRUD Operations

Create

Read

Update

Delete

HTTP Request Methods

CRUD

HTTP Request Methods

Create

POST

Read

GET

Update

PUT

Delete

DELETE

GET HTTP Request Method



Creating a FastAPI application

File: books.py

```
from fastapi import FastAPI  
  
app = FastAPI()  
  
@app.get("/api-endpoint")  
async def first_api():  
    return {'message': 'Hello Eric!'}
```

Lets dive into the first function

Dive in

Read

Get

API Endpoint:

```
@app.get("/api-endpoint")
async def first_api():
    return {'message': 'Hello Eric!'}
```

URL : 127.0.0.1:8000/api-endpoint

Response:

```
{
    "message": "Hello Eric!"
}
```

Start FastAPI Application

Read

Get

API Endpoint:

```
@app.get("/api-endpoint")
async def first_api():
    return {'message': 'Hello Eric!'}
```

Run FastAPI application:

```
ericroby@Erics-MacBook-Pro ~% uvicorn books:app --reload
```

URL : 127.0.0.1:8000

Change Endpoint for Books

Read

Get

```
BOOKS = {  
    {'title': 'Title One', 'author': 'Author One', 'category': 'science'},  
    {'title': 'Title Two', 'author': 'Author Two', 'category': 'science'},  
    {'title': 'Title Three', 'author': 'Author Three', 'category': 'history'},  
    {'title': 'Title Four', 'author': 'Author Four', 'category': 'math'},  
    {'title': 'Title Five', 'author': 'Author Five', 'category': 'math'},  
}  
RETURN BOOKS
```

URL : URL:0.0.0.0:8000/api/v1/books

Path Parameters

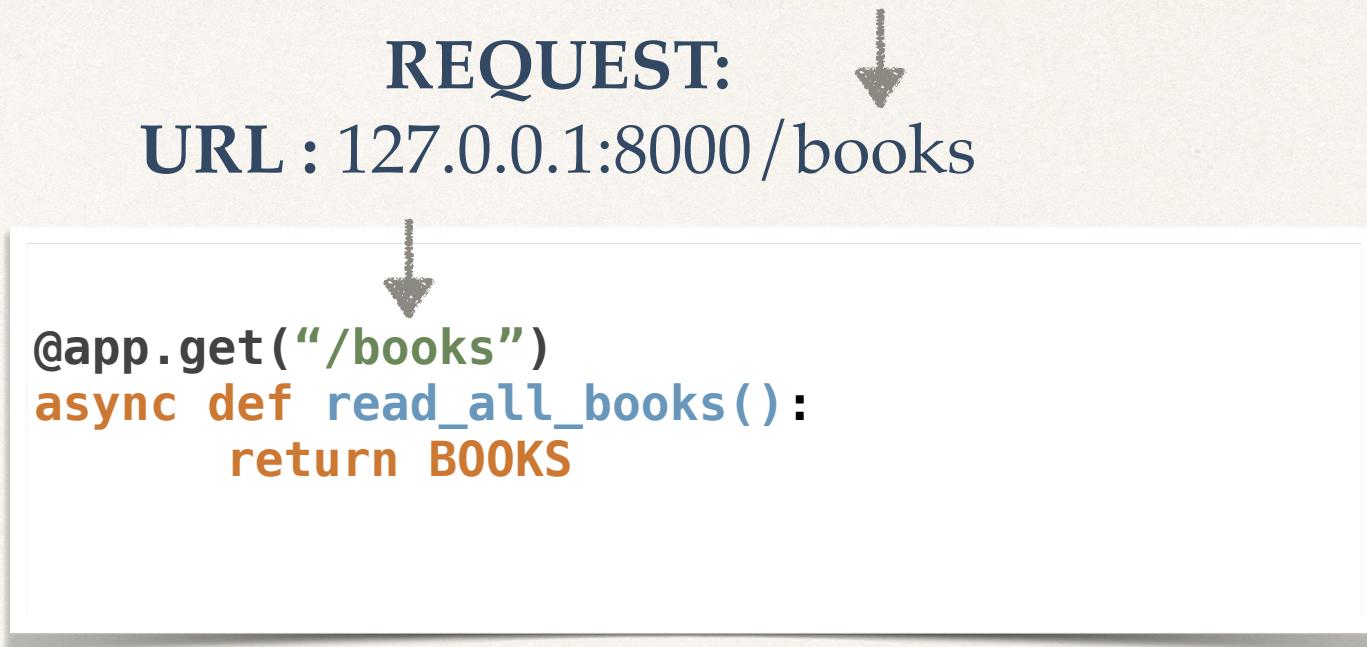
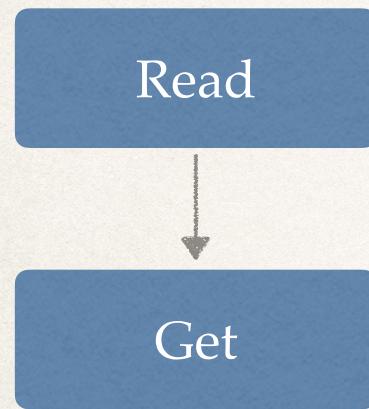


What are Path Parameters

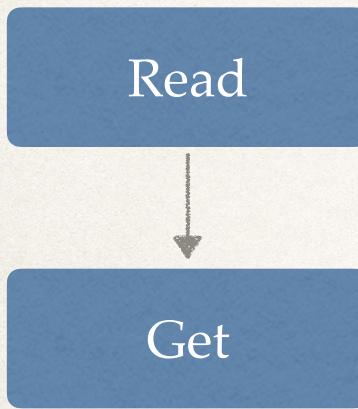
- Path Parameters are request parameters that have been attached to the URL
- Path Parameters are usually defined as a way to find information based on location
- Think of a computer file system:
 - You can identify the specific resources based on the file you are in

```
/Users/codingwithroby/Documents/python/fastapi/section1
```

Path Parameters



Path Parameters



REQUEST:

URL : 127.0.0.1:8000/books/book_one

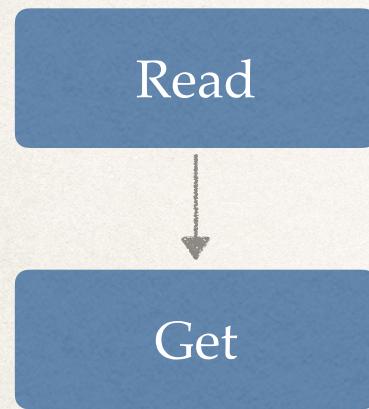


```
@app.get("/books/{dynamic_param}")
async def read_all_books(dynamic_param):
    return {'dynamic_param': dynamic_param}
```

RESPONSE:

```
{
    "dynamic_param": "book_one"
}
```

Order Matters with Path Parameters



REQUEST:

URL : 127.0.0.1:8000/books/mybook

```
@app.get("/books/mybook")
async def read_all_books():
    return {'book_title': 'My Favorite Book'}
```



```
@app.get("/books/{dynamic_param}")
async def read_all_books(dynamic_param):
    return {'dynamic_param': dynamic_param}
```

Path Parameters

Read

↓
Get

REQUEST:

URL : 127.0.0.1:8000/books/ = title four

```
BOOKS = [
    {'title': 'Title One', 'author': 'Author One', 'category': 'science'},
    {'title': 'Title Two', 'author': 'Author Two', 'category': 'science'},
    {'title': 'Title Three', 'author': 'Author Three', 'category': 'history'},
    {'title': 'Title Four', 'author': 'Author Four', 'category': 'math'},
    {'title': 'Title Five', 'author': 'Author Five', 'category': 'math'},
]
```

```
@app.get("/books/{book_title}")
async def read_book(book_title: str):
    for book in BOOKS:
        if book.get('title').casifold() ==
            book_title.casifold():
```

RESPONSE:

```
{
    "title": "Title Four",
    "author": "Author Four",
    "category": "math"
}
```

Query Parameters



What are Query Parameters

- **Query Parameters are request parameters that have been attached after a “?”**
- **Query Parameters have name=value pairs**
- **Example:**
 - `127.0.0.1:8000/books/?category=math`

Query Parameters

REQUEST:

URL : 127.0.0.1:8000/books/?category=science



```
@app.get("/books/")
async def read_category_by_query(category: str):
    books_to_return = []
    for book in BOOKS:
        if book.get('category').casifold() == category.casifold():
            books_to_return.append(book)

    return books_to_return
```

Query Parameters

REQUEST:

URL : 127.0.0.1:8000/books/author%20four/?category=science

```
@app.get("/books/{book_author}/")
async def read_category_by_query(book_author: str, category: str):
    books_to_return = []
    for book in BOOKS:
        if book.get('author').casfold() == book_author.casfold() and
            book.get('category').casfold() == category.casfold():
            books_to_return.append(book)

    return books_to_return
```

POST HTTP Request Method

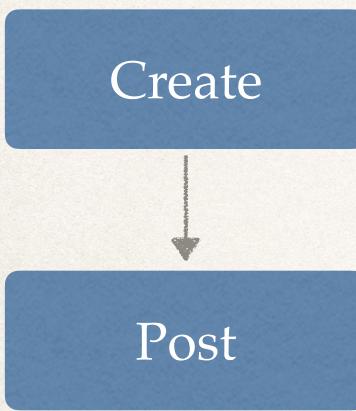


What is the POST Request Method

- Used to create data
- POST can have a body that has additional information that GET does not have
- Example of Body:

```
{"title": "Title Seven", "author": "Author Two", "category":
```

POST Request Method



REQUEST:

URL : 127.0.0.1:8000/books/create_book

```
graph TD; Post --> Callout[ ]; Callout --> Code[@app.post("/books/create_book")\nasync def create_book(new_book=Body()):\n    BOOKS.append(new_book)]
```

A diagram illustrating a POST request. On the left, a blue rounded rectangle labeled "Post" has an arrow pointing down to a larger white callout box. Inside the callout box is another arrow pointing down to a code snippet. The code is written in Python and defines a POST endpoint for creating a book. It uses the FastAPI framework's annotations: `@app.post`, `new_book=Body()`, and `BOOKS.append`.

PUT HTTP Request Method



What is the PUT Request Method

- Used to update data
- PUT can have a body that has additional information (like POST) that GET does not have
- Example of Body:

```
{"title": "Title Six", "author": "Author Two", "category":
```

PUT Request Method

Update



Put

REQUEST:

URL : 127.0.0.1:8000/books/update_book

```
@app.put("/books/update_book")
async def update_book(updated_book=Body()):
    for i in range(len(BOOKS)):
        if BOOKS[i].get('title').casfold() == updated_book.get('title').casfold():
            BOOKS[i] = updated_book
```

DELETE HTTP Request Method



What is the DELETE Request Method

- Used to delete data

Delete

Delete

REQUEST:

URL : 127.0.0.1:8000/books/delete_book/{book_title}

```
@app.delete("/books/delete_book/{book_title}")
async def delete_book(book_title: str):
    for i in range(len(BOOKS)):
        if BOOKS[i].get('title').casfold() == book_title.casfold():
            BOOKS.pop(i)
            break
```

Books 2 Project



Project 2

- Project two will still be focused on creating Book API Endpoints
- Continued Education includes:
 - GET, POST, PUT, DELETE Request Methods
- New Information will include:
 - Data Validation, Exception Handling, Status Codes, Swagger Configuration, Python Request Objects

Creating a new books project

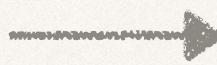
- We will create a new class called Book
- We will be using these books throughout this project.

```
class Book:  
    id: int  
    title: str  
    author: str  
    description: str  
    rating: int  
  
def __init__(self, id, title, author, description, rating):  
    self.id = id  
    self.title = title  
    self.author = author  
    self.description = description  
    self.rating = rating
```

HTTP Request Methods

CRUD

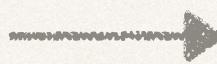
Create



HTTP Request Methods

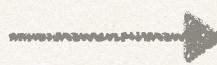
POST

Read



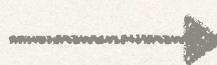
GET

Update



PUT

Delete



DELETE

Pydantics



What is Pydantics

- Python library that is used for data modeling, data parsing and has efficient error handling.
- Pydantics is commonly used as a resource for data validation and how to handle data coming to our FastAPI application.

We will be implementing Pydantics

- Create a different request model for data validation
- Field data validation on each variable / element



```
class BookRequest(BaseModel):  
    id: int  
    title: str = Field(min_length=3)  
    author: str = Field(min_length=1)  
    description: str = Field(min_length=1, max_length=100)  
    rating: int = Field(gt=0, lt=5)
```

BookRequest and Book Conversion

- We will convert the Pydantics Request into a Book
- Here is an example within an upcoming Post Request Method

```
@app.post("/create-book")
async def create_book(book_request: BookRequest):
    new_book = Book(**book_request.dict())
    BOOKS.append(new_book)
```

** operator will pass the key/value from BookRequest() into the Book() constructor

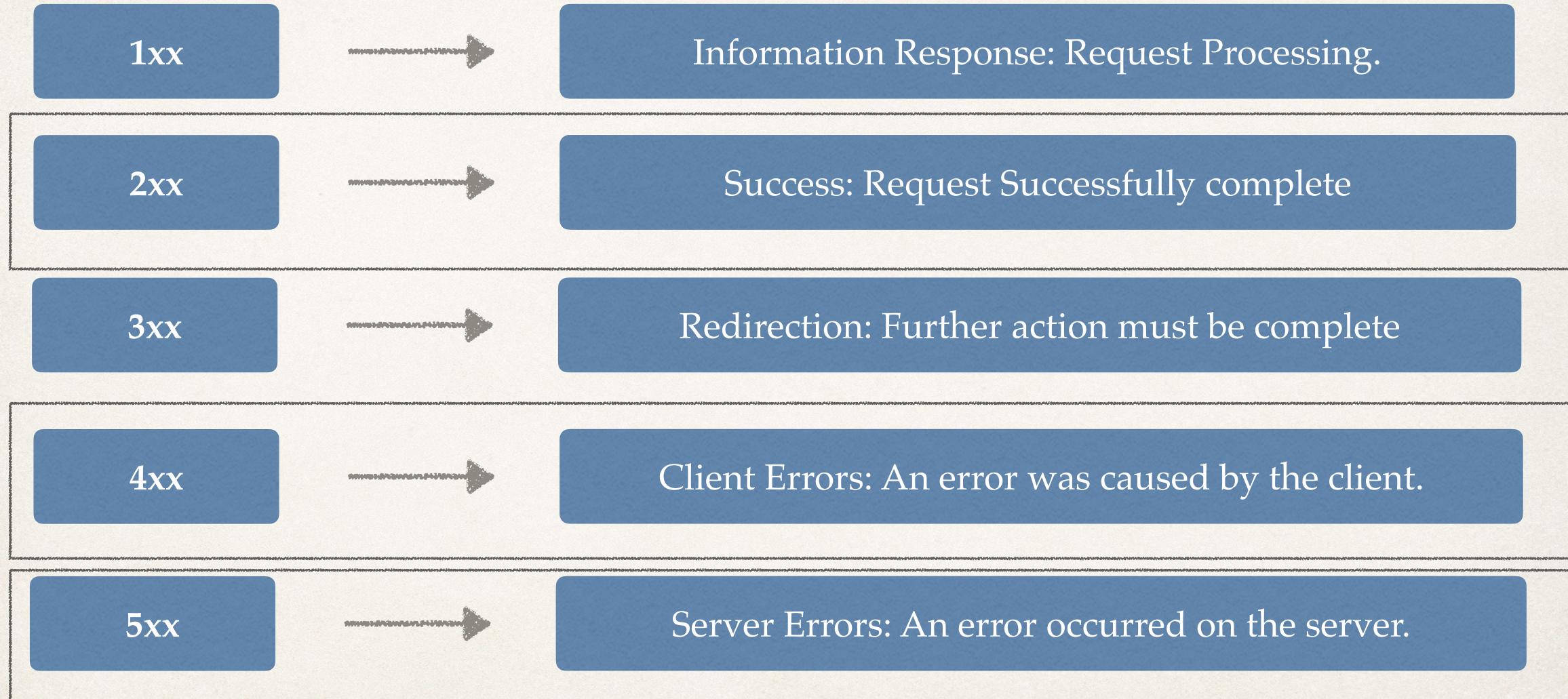
Status Codes



What are Status Codes?

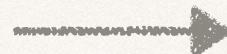
- An HTTP Status Code is used to help the Client (the user or system submitting data to the server) to understand what happened on the server side application.
- Status Codes are international standards on how a Client/Server should handle the result of a request.
- It allows everyone who sends a request to know if their submission was successful or not.

Status Codes:



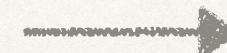
2xx Successful Status Codes:

200: OK



Standard Response for a Successful Request. Commonly used for successful Get requests when data is being returned.

201:



The request has been successful, creating a new resource. Used when a POST creates an entity.

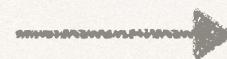
204: No



The request has been successful, did not create an entity nor return anything. Commonly used with PUT requests.

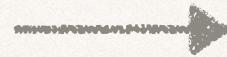
4xx Client Errors Status Codes:

400: Bad



Cannot process request due to client error. Commonly used for invalid request methods.

401: Unauthorized



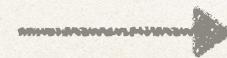
Client does not have valid authentication for target resource

404: Not



The clients requested resource can not be found

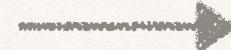
422: Unprocessable Entity



Semantic Errors in Client Request

5xx Server Status Codes:

500: Internal Server
Error



Generic Error Message, when an unexpected issue on the server happened.

Project 3



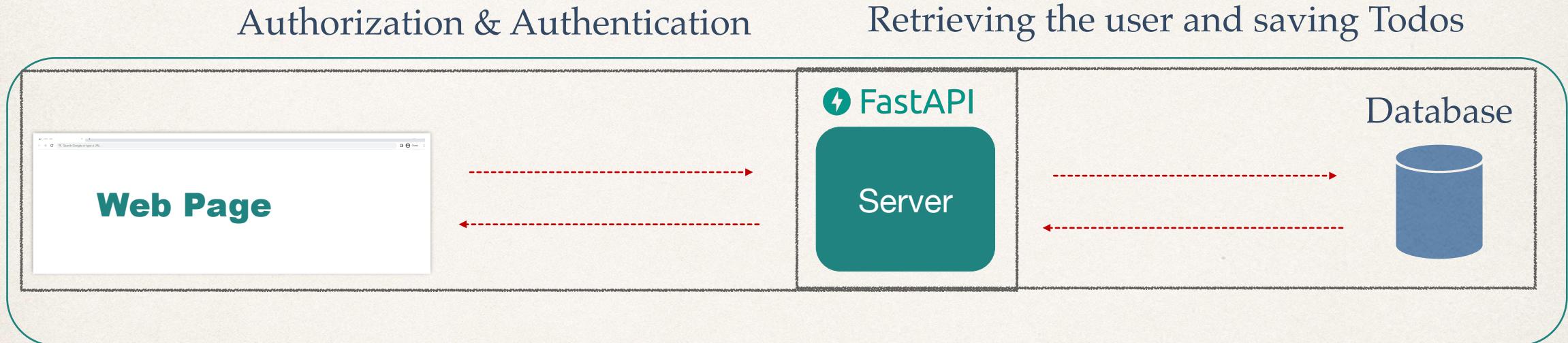
Project 3

- Project three we will be switching our focus to TODOS instead of BOOKS
- New Information will include:
 - Full SQL Database
 - Authentication
 - Authorization
 - Hashing Passwords

Continued learning from
other projects

Creating a Todo Table

- We will create new Todo Table Models for our application
- We will be using these Todos to save records throughout this project



SQL Database Introduction



What is a database?

- Organized collection of structured information of **data**, which is stored in a computer system.
- The data can be easily accessed.
- The data can be modified.
- The data can be combined.
- Many databases use a structured query language (SQL) to modify and write data

What is Data?

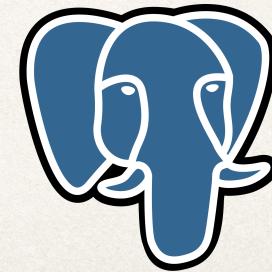
What is a database?

- Data can be related to just about any object.
- For example, a user on an application may have:
 - Name
 - Age
 - Email
 - Password

All of this is Data

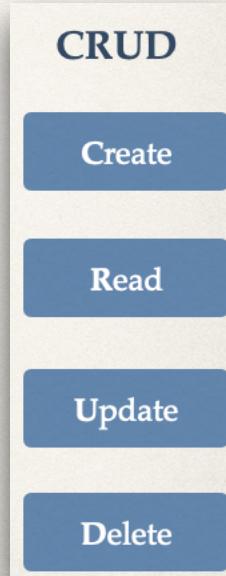
What is a database?

- A database is a collection of data
- Since data, on its own, is just data. A database allows management of this data
- Databases are organized in how data can be retrieved, stored and modified
- There are many types of Database Management Systems (DBMS)

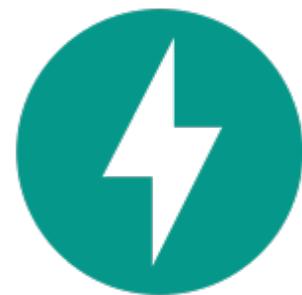


What is a SQL?

- Pronounced either as S-Q-L or “See Quel”
- Standard language for dealing with relational databases
- SQL can be used to do different things with database records:
 - Create
 - Read
 - Update
 - Delete



JSON WEB TOKEN (JWT) OVERVIEW



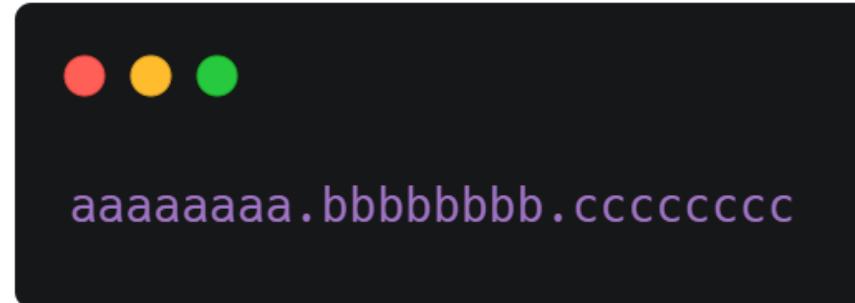
FastAPI

WHAT IS A JSON WEB TOKEN?

- JSON Web Token is a self-contained way to securely transmit data and information between two parties using a JSON Object.
- JSON Web Tokens can be trusted because each JWT can be digitally signed, which in return allows the server to know if the JWT has been changed at all
- JWT should be used when dealing with authorization
- JWT is a great way for information to be exchanged between the server and client

JSON WEB TOKEN STRUCTURE

- A JSON Web Token is created of three separate parts separated by dots (.) which include:
 - Header : (a)
 - Payload : (b)
 - Signature : (c)



JWT HEADER

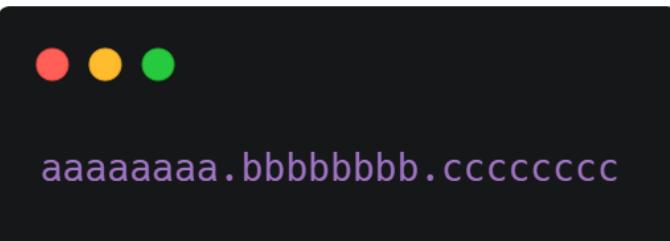
- A JWT header usually consists of two parts:
 - (alg) The algorithm for signing
 - “typ” The specific type of token
 - The JWT header is then encoded using Base64 to create the first part of the JWT (a)

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

A screenshot of a mobile application interface. At the top left, there are three colored dots: red, yellow, and green. Below them is a large, empty text input field containing the placeholder text "aaaaaaaa.bbbbbbbb.cccccccc".

JWT PAYLOAD

- A JWT Payload consists of the data.
The Payloads data contains claims, and there are three different types of claims.
 - Registered
 - Public
 - Private
- The JWT Payload is then encoded using Base64 to create the second part of the JWT
(b)



```
{  
  "sub": "1234567890",  
  "name": "Eric Roby",  
  "given_name": "Eric",  
  "family_name": "Roby",  
  "email": "codingwithrobby@gmail.com"  
  "admin": true  
}
```

JWT SIGNATURE

- A JWT Signature is created by using the algorithm in the header to hash out the encoded header, encoded payload with a secret.
- The secret can be anything, but is saved somewhere on the server that the client does not have access to
- The signature is the third and final part of a JWT (c)



aaaaaaaa.bbbbbbbb.cccccccc



```
HMACSHA256(  
    base64UrlEncode(header) + "." +  
    base64UrlEncode(payload),  
    secret)
```

JWT EXAMPLE

JWT Header

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

JWT Payload

```
{  
  "sub": "1234567890",  
  "name": "Eric Roby",  
  "given_name": "Eric",  
  "family_name": "Roby",  
  "email": "codingwithroby@gmail.com"  
  "admin": true  
}
```

JWT Signature

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  learnonline)
```

JSON Web Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkVyaW MgUm9ieSIsImdpdmVuX25hbWUiOijFcmljIiwiZmFtaWx5X25hbWUiOijSb2J5IiwiZW1haWwiOijjb2Rp bmd3aXRocm9ieUBnbWFpbC5jb20iLCJhZG1pbiiI6dHJ1ZX0.i8yqz0z-nWr5hwXqRYP18W9igUPoKMZiBZW315tK5g8
```

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIi0iIxMjM0NTY3ODkwIiwibmFtZSI6IkVyaWMgUm9ieSIsImdpdmVuX25hbWUi0iJFcmljIiwiZmFtaWx5X25hbWUi0iJSb2J5IiwiZW1haWwi0iJjb2Rpbmd3aXRo cm9ieUBnbWFpbC5jb20iLCJhZG1pbii6dHJ1ZX0.i8yqz0z-nWr5hwXqRYP18W9igUPoKMZiBZW315tK5g8
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "sub": "1234567890",  
  "name": "Eric Roby",  
  "given_name": "Eric",  
  "family_name": "Roby",  
  "email": "codingwithroby@gmail.com",  
  "admin": true  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  learnonline  
)  secret base64 encoded
```

<https://jwt.io>

⌚ Signature Verified

SHARE JWT

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJdWIi0iIxMjM0NTY3ODkwIiwibmFtZSI6IkVyaWMgUm9ieSIsImdpdmVuX25hbWUiOiJFcmljIiwiZmFtaWx5X25hbWUiOiJSb2J5IiwiZW1haWwiOiJjb2Rpmd3aXRocm9ieUBnbWFpbC5jb20iLCJhZG1ppiI6dHJ1ZX0.i8yqz0z-nWr5hwXqRYP18W9igUPoKMZiBZW315tK5g|
```

Decoded

EDIT THE PAYLOAD AND SECRET

<https://jwt.io>

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

```
{  
  "sub": "1234567890",  
  "name": "Eric Roby",  
  "given_name": "Eric",  
  "family_name": "Roby",  
  "email": "codingwithroby@gmail.com",  
  "admin": true  
}
```

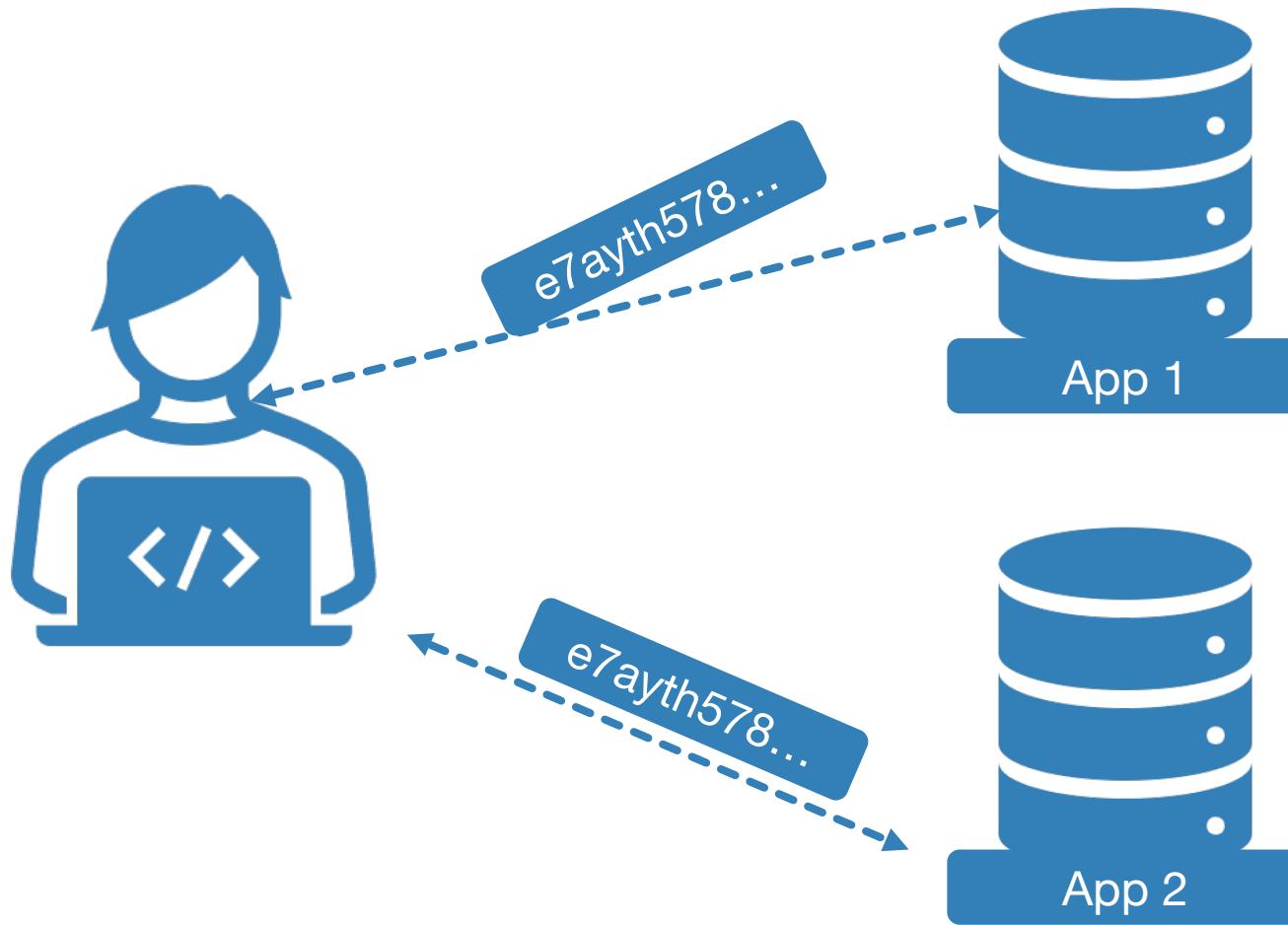
VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  learnonline  
)  secret base64 encoded
```

⊗ Invalid Signature

SHARE JWT

JWT PRACTICAL USE CASE



PRODUCTION DATABASE INTRODUCTION



FastAPI

PRODUCTION DATABASE

- This section will go over installing a production

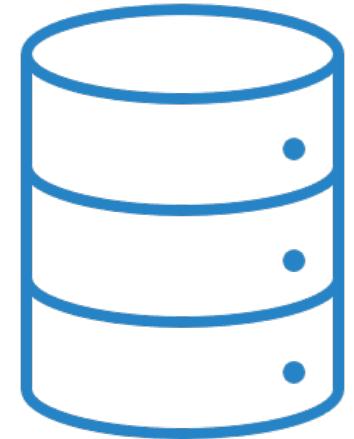
Before jumping in, lets take
about the difference between
SQLite and production DBMS

either one



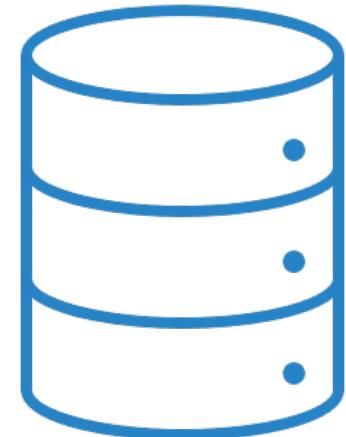
PRODUCTION DBMS VS SQLITE3

- SQLite3 strives to provide local data storage for individual applications and devices.
- SQLite3 emphasizes economy, efficiency and simplicity.
- For most small / medium applications, SQLite3 will work perfectly.
- SQLite3 focuses on different concepts than a production Database Management System.



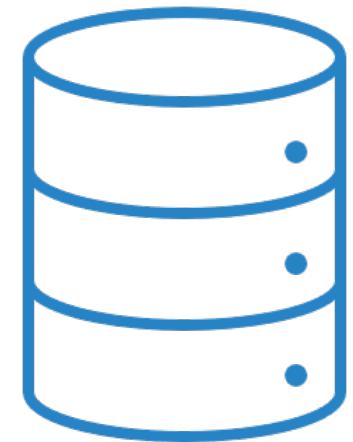
PRODUCTION DBMS VS SQLITE3

- MySQL & PostgreSQL focuses on a big difference compared to SQLite3.
- These production DBMS focuses on scalability, concurrency and control.
- If your application is going to have 10s of thousands of users, it may be wise to switch to a production DBMS
- If your application is only you, and a few others, SQLite3 will work great!



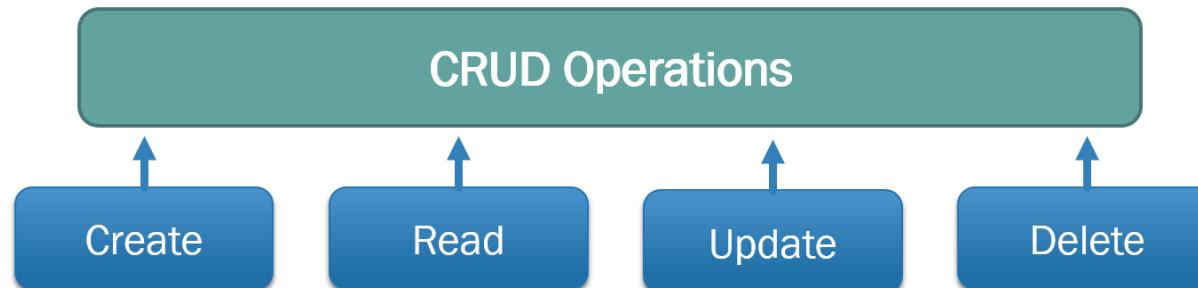
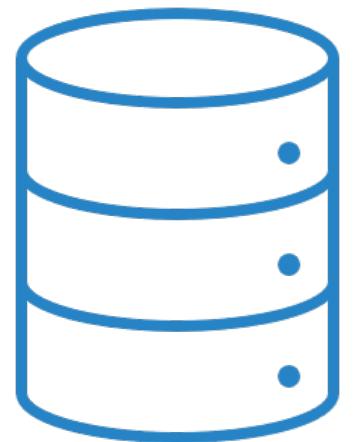
PRODUCTION DBMS KEY NOTES:

- SQLite3 runs in-memory, which allows development of a SQLite3 data to be easy, as it is part of your application!
- Production DBMS run on their own server and port. Which means you need to make sure the database is running, and have authentication linking to the DBMS
- (SQLite3) For deployment you can deploy a SQLite3 database along with the application
- (Prod DBMS) For deployment you will need to also deploy the database separate from the application

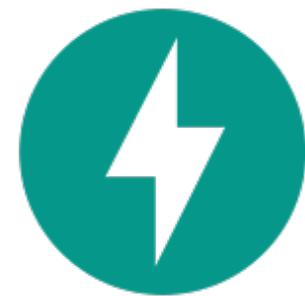


OVERVIEW OF SECTION (FOR BOTH MYSQL & POSTGRESQL)

- We will install the production DBMS
- Setup the tables and data within the production DBMS
- Connect the production DBMS to our application
- Push data from application to our production DBMS!



BASIC SQL QUERIES



FastAPI

INSERTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete	owner (FK)

DELETE THIS IS THE USERS TABLE

Id (PK)	email	username	first_name	last_name	hashed_password	is_active

DELETE THIS IS THE USERS TABLE

Id (PK)	email	username	first_name	last_name	hashed_password	is_active
1	codingwithroby@gmail.com	codingwithroby	Eric	Roby	123abcEqw!..	1
2	exampleuser12@example.com	exampleuser	Example	User	Gjjjd!2!..	1

INSERTING DATABASE TABLE (TODOS)

```
INSERT INTO todos (title,  
description, priority, complete)  
  
VALUES ('Go to store', 'To pick up  
eggs' 4, False);
```

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0

INSERTING DATABASE TABLE (TODOS)

```
INSERT INTO todos (title,  
description, priority, complete)  
  
VALUES ('Haircut', 'Need to get  
length 1mm' 3, False);
```

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0

STARTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0

STARTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0

STARTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0
4	Water plant	Inside and Outside plants	4	0

STARTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0
4	Water plant	Inside and Outside plants	4	0
5	Learn something new	Learn to program	5	0

STARTING DATABASE TABLE (TODOS)

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0
4	Water plant	Inside and Outside plants	4	0
5	Learn something new	Learn to program	5	0
6	Shower	You have not showered in days	5	0

DELETE

Id (PK)	title	description	priority	complete	owner
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
6	Shower	You have not showered in days	5	0	2

SELECT SQL QUERIES

```
SELECT * FROM todos;
```



Select ALL columns and rows

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0
4	Water plant	Inside and Outside plants	4	0
5	Learn something new	Learn to program	5	0
6	Shower	You have not showered in days	5	0

SELECT SQL QUERIES

```
SELECT title FROM todos;
```



Select just title from columns

title
Go to store
Haircut
Feed dog
Water plant
Learn something new
Shower

SELECT SQL QUERIES

```
SELECT description FROM todos;
```

Select just description from columns

description
To pick up eggs
Need to get length 1mm
Make sure to use new food brand
Inside and Outside plants
Learn to program
You have not showered in days

SELECT SQL QUERIES

```
SELECT title, description FROM  
todos;
```

Select title, description from columns

title	description
Go to store	To pick up eggs
Haircut	Need to get length 1mm
Feed dog	Make sure to use new food brand
Water plant	Inside and Outside plants
Learn something new	Learn to program
Shower	You have not showered in days

SELECT SQL QUERIES

```
SELECT title, description, priority FROM  
todos;
```

Select title, description and priority from columns

title	description	priority
Go to store	To pick up eggs	4
Haircut	Need to get length 1mm	3
Feed dog	Make sure to use new food brand	5
Water plant	Inside and Outside plants	4
Learn something new	Learn to program	5
Shower	You have not showered in days	5

WHERE SQL QUERIES

WHERE Clause

WHERE SQL QUERIES

```
SELECT * FROM todos WHERE priority=5;
```

Select ALL rows & columns WHERE priority = 5

Id (PK)	title	description	priority	complete
3	Feed dog	Make sure to use new food brand	5	0
5	Learn something new	Learn to program	5	0
6	Shower	You have not showered in days	5	0

WHERE SQL QUERIES

```
SELECT * FROM todos WHERE title='Feed dog';
```

Select ALL rows & columns WHERE title= Feed dog

Id (PK)	title	description	priority	complete
3	Feed dog	Make sure to use new food brand	5	0

WHERE SQL QUERIES

```
SELECT * FROM todos WHERE id=2;
```

Select ALL rows & columns WHERE id= 2

Id (PK)	title	description	priority	complete
2	Haircut	Need to get length 1mm	3	0

UPDATE SQL QUERIES

UPDATE Clause

WHERE SQL QUERIES

```
UPDATE todos SET complete=True WHERE  
title='Learn something new';
```

Update ALL rows & columns to now have complete = True
WHERE id = 5

Id (PK)	title	description	priority	complete
5	Learn something new	Learn to program	5	1

1 == True

WHERE SQL QUERIES

```
UPDATE todos SET complete=True WHERE id=5;
```

Update ALL rows & columns to now have complete = True
WHERE id = 5

Id (PK)	title	description	priority	complete
5	Learn something new	Learn to program	5	1

1 == True

DELETE SQL QUERIES

DELETE Clause

DELETE SQL QUERIES

```
DELETE FROM todos WHERE id=5;
```

Delete ALL rows and columns where id =
5

Id (PK)	title	description	priority	complete
1	Go to store	To pick up eggs	4	0
2	Haircut	Need to get length 1mm	3	0
3	Feed dog	Make sure to use new food brand	5	0
4	Water plant	Inside and Outside plants	4	0
6	Shower	You have not showered in days	5	0

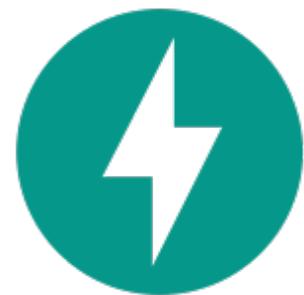
DELETE SQL QUERIES

```
DELETE FROM todos WHERE complete=0;
```

Delete ALL rows and columns where complete =
0

Id (PK)	title	description	priority	complete

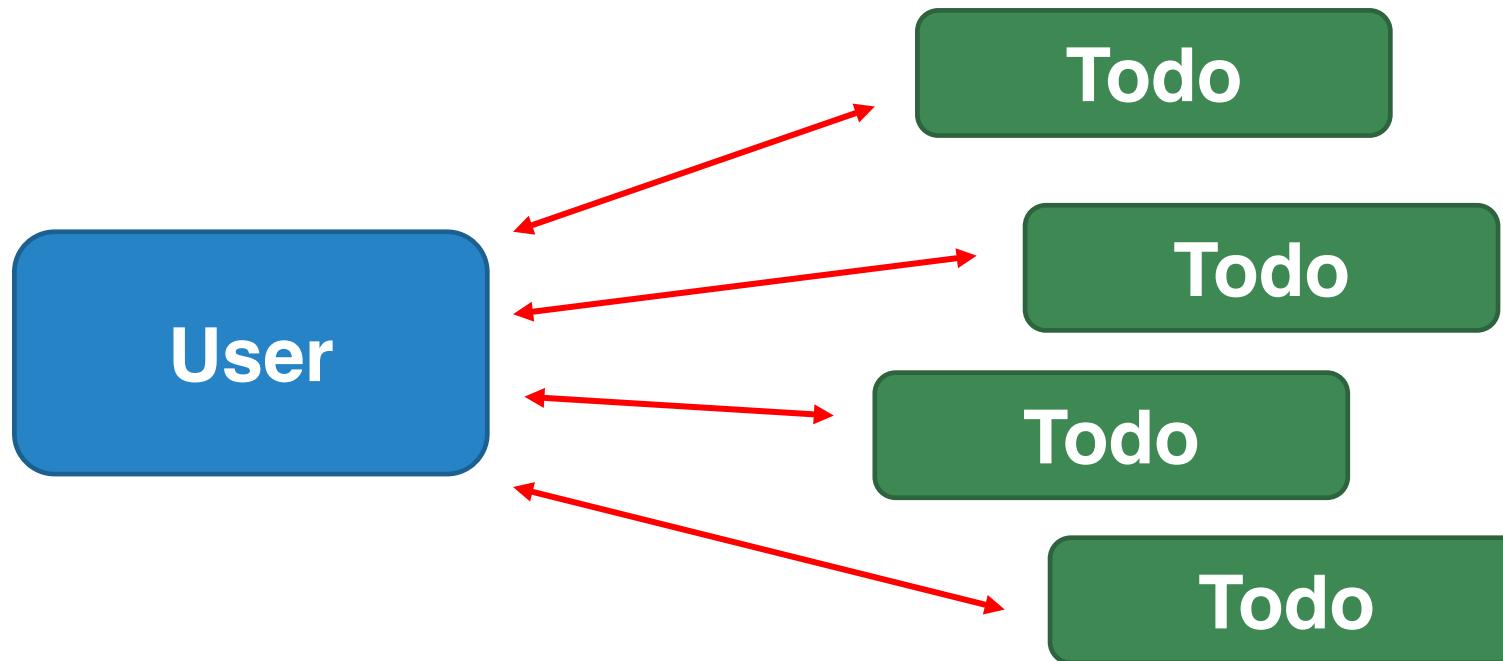
ONE TO MANY INTRODUCTION



FastAPI

WHAT IS A ONE TO MANY RELATIONSHIP

- A user can have many todos



WHAT IS A ONE TO MANY RELATIONSHIP

Users & Todos



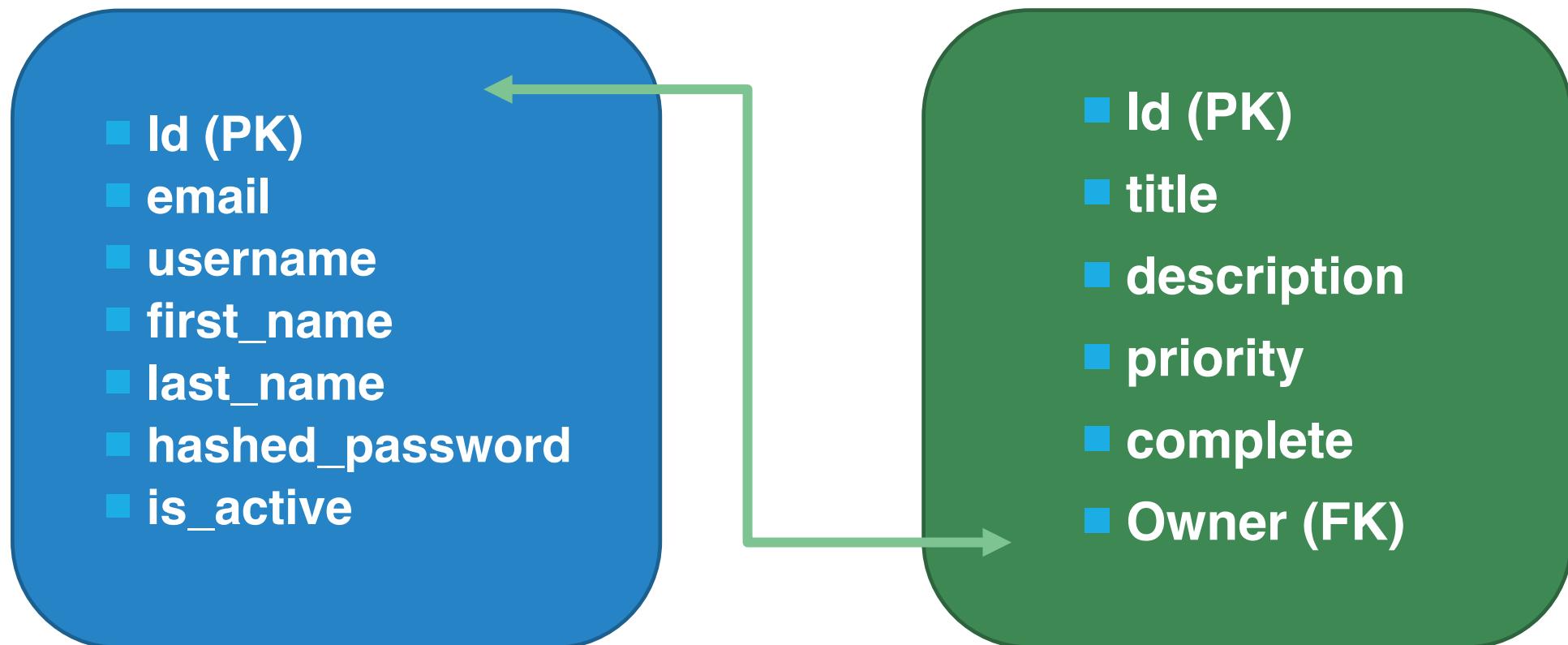
Id (PK)	email	username	first_name	last_name	hashed_password	is_active



Id (PK)	title	description	priority	complete

WHAT IS A ONE TO MANY RELATIONSHIP

Users & Todos



WHAT IS A ONE TO MANY RELATIONSHIP

Id (PK)	title	description	priority	complete



Id (PK)	title	description	priority	complete	owner (FK)

WHAT IS A ONE TO MANY RELATIONSHIP

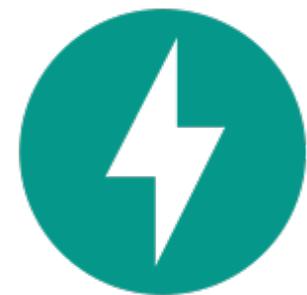
Users

Id (PK)	email	username	first_name	last_name	hashed_password	is_active
1	codingwithrobby@gmail.com	codingwithrobby	Eric	Roby	123abcEqwl..	1
2	exampleuser12@example.com	exampleuser	Example	User	Gjjjd!2!..	1

Todos

Id (PK)	title	description	priority	complete	owner (FK)
1	Go to store	To pick up eggs	4	0	1
2	Haircut	Need to get length 1mm	3	0	1
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
5	Learn something new	Learn to program	5	0	1
6	Shower	You have not showered in days	5	0	2

FOREIGN KEYS



FastAPI

WHAT IS A FOREIGN KEY?

- A foreign key (FK) is a column within a relational database table that provides a link between two separate tables
- A foreign key references
- Most relational databases tables together to present

The diagram illustrates a foreign key relationship between the **Users** and **Todos** tables. A yellow arrow points from the **email** column in the **Users** table to the **owner (FK)** column in the **Todos** table, indicating that the **email** column in **Users** serves as a foreign key linking to the **owner (FK)** column in **Todos**.

Users

Id (PK)	email	username	first_name	last_name	hashed_password	is_active
1	codingwithrob...@gmail.com	codingwithrobby	Eric	Roby	123abcEqwl...	1
2	exampleuser12...@example.com	exampleuser	Example	User	Gjjjd!2l...	1

Todos

Id (PK)	title	description	priority	complete	owner (FK)
1	Go to store	To pick up eggs	4	0	1
2	Haircut	Need to get length 1mm	3	0	1
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
5	Learn something new	Learn to program	5	0	1
6	Shower	You have not showered in days	5	0	2

FOREIGN KEYS

```
SELECT * FROM todos;
```



Select ALL columns and rows

Id (PK)	title	description	priority	complete	owner (FK)
1	Go to store	To pick up eggs	4	0	1
2	Haircut	Need to get length 1mm	3	0	1
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
5	Learn something new	Learn to program	5	0	1
6	Shower	You have not showered in days	5	0	2

FOREIGN KEYS

```
SELECT * FROM users;
```



Select ALL columns and rows

Id (PK)	email	username	first_name	last_name	hashed_password	is_active
1	codingwithroby@gmail.com	codingwithroby	Eric	Roby	123abcEqw!..	1
2	exampleuser12@example.com	exampleuser	Example	User	Gjjjd!2!..	1

- Each API request a user will have their ID attached
 - If we have the user ID attached to each request, we can use the ID to find their todos

Id (PK)	email	username	first_name	last_name	hashed_password	is_active
1	codingwithroby@gmail.com	codingwithroby	Eric	Roby	123abcEqw!..	1
2	exampleuser12@example.com	exampleuser	Example	User	Gjjjd!2!..	1

Id (PK)	title	description	priority	complete	owner (FK)
1	Go to store	To pick up eggs	4	0	1
2	Haircut	Need to get length 1mm	3	0	1
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
5	Learn something new	Learn to program	5	0	1
6	Shower	You have not showered in days	5	0	2

WHERE SQL QUERIES

```
SELECT * FROM todos WHERE owner=1;
```

Select ALL rows & columns WHERE owner = 1

Id (PK)	title	description	priority	complete	owner
1	Go to store	To pick up eggs	4	0	1
2	Haircut	Need to get length 1mm	3	0	1
5	Learn something new	Learn to program	5	0	1

WHERE SQL QUERIES

```
SELECT * FROM todos WHERE owner=2;
```

Select ALL rows & columns WHERE owner = 1

Id (PK)	title	description	priority	complete	owner
3	Feed dog	Make sure to use new food brand	5	0	2
4	Water plant	Inside and Outside plants	4	0	2
6	Shower	You have not showered in days	5	0	2

MYSQL INTRODUCTION



FastAPI

WHAT IS MYSQL

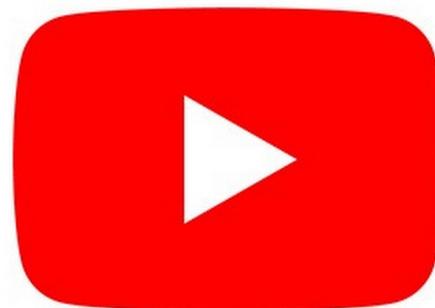
- Open-source relational database management system
- Requires a server
- Production ready
- Scalable
- Secure



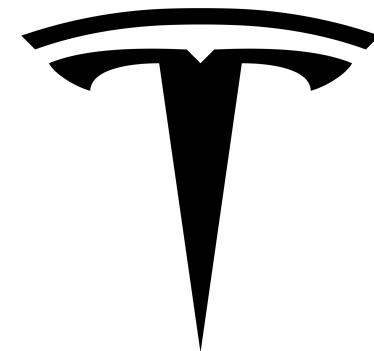
WHO USES/USED MYSQL?



Facebook

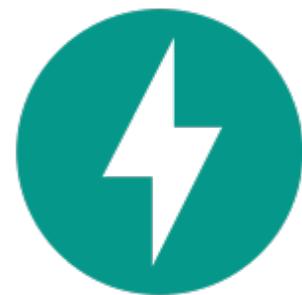


YouTube



Tesla

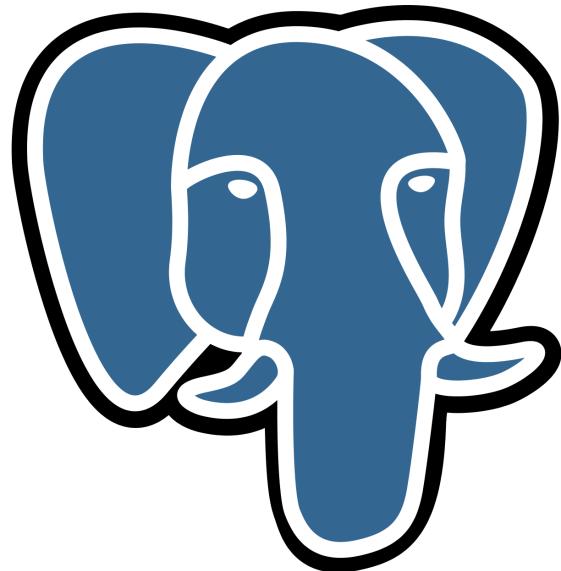
POSTGRESQL INTRODUCTION



FastAPI

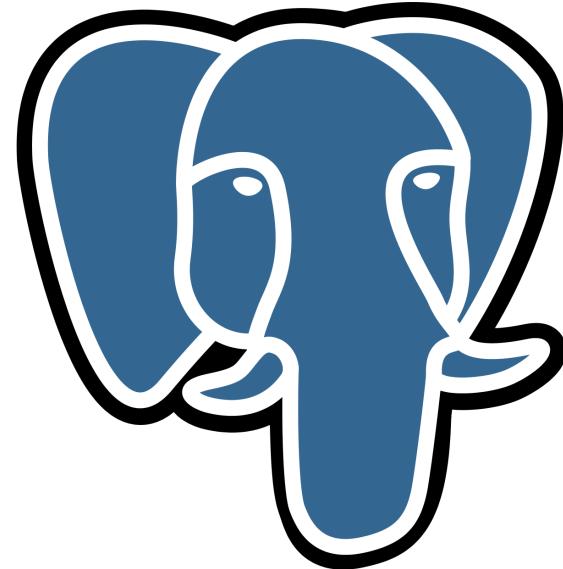
WHAT IS POSTGRESQL

- Production ready
- Open-source relational database management system
- Secure
- Requires a server
- Scalable



WHAT WILL WE COVER?

- How to install PostgreSQL
 - Windows
 - Mac
- Setup SQL tables
- Connect PostgreSQL to our application



WHO USES/USED POSTGRESQL?



App
le

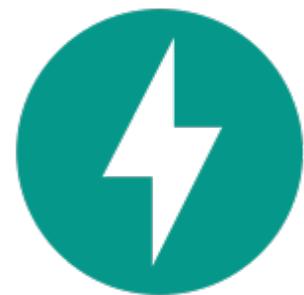


Reddit



Twitch

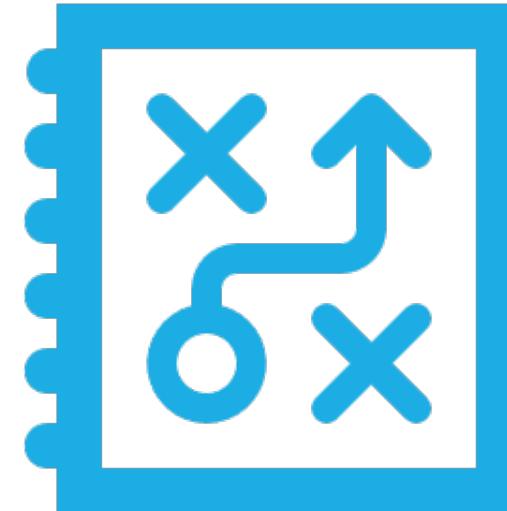
ROUTING INTRODUCTION



FastAPI

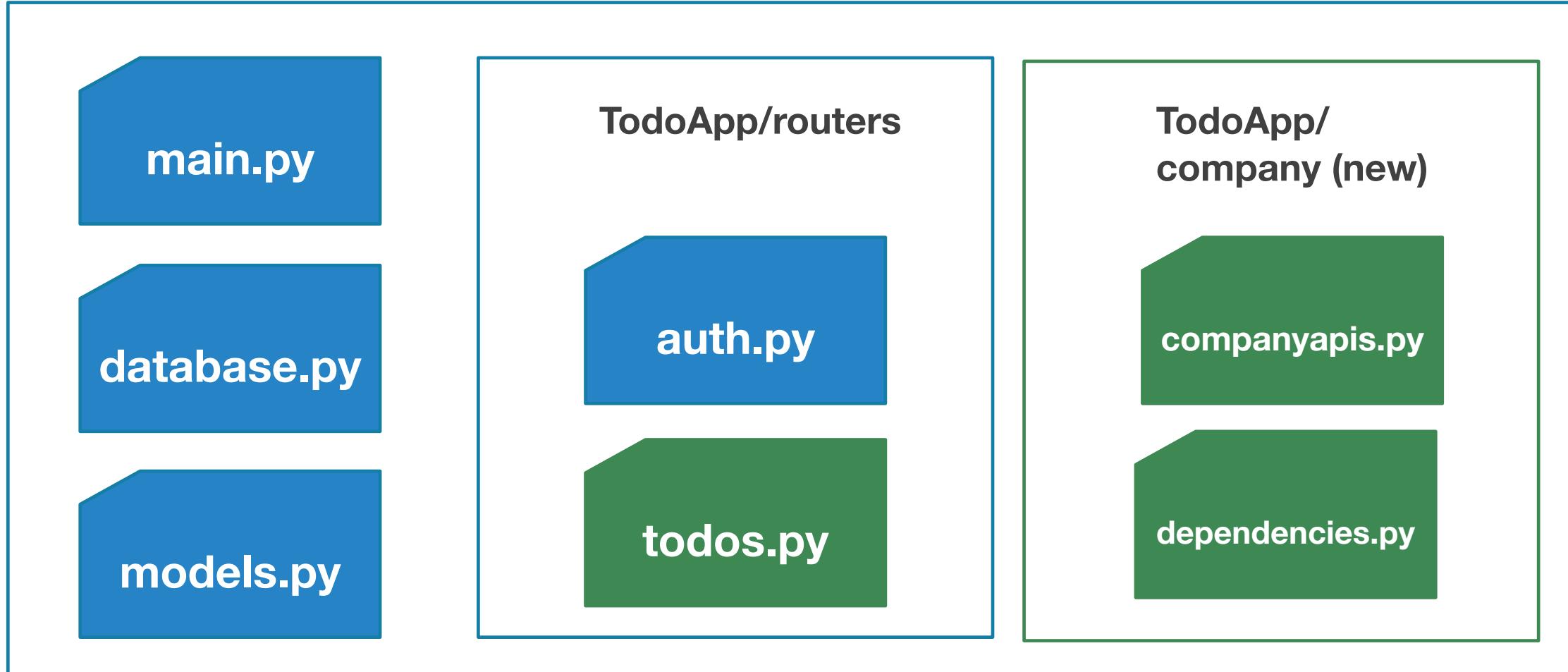
WHAT IS ROUTING

- Rare that you want your entire application to be on a single file
- Flexible tool to structure your application
- Scalable architecture
- Organize file structure



NEW PROJECT STRUCTURE

TodoApp



auth



POST

/auth/create/user Create New User



POST

/auth/token Login For Access Token



todos



GET

/todos/ Read All



POST

/todos/ Create Todo



GET

/todos/user Read All By User



GET

/todos/{todo_id} Read Todo



PUT

/todos/{todo_id} Update Todo



DELETE

/todos/{todo_id} Delete Todo



companyapis



GET

/companyapis/ Get Company Name

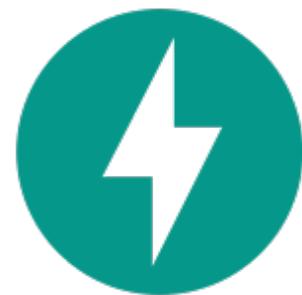


GET

/companyapis/employees Number Of Employees



JINJA SCRIPTS



FastAPI

WHAT JINJA?

- Fast, expressive and extensible templating language
- Able to write code similar to Python in the DOM
- The template is passed data to render within the final document



WHAT ARE JINJA TEMPLATING TAGS AND SCRIPTS?

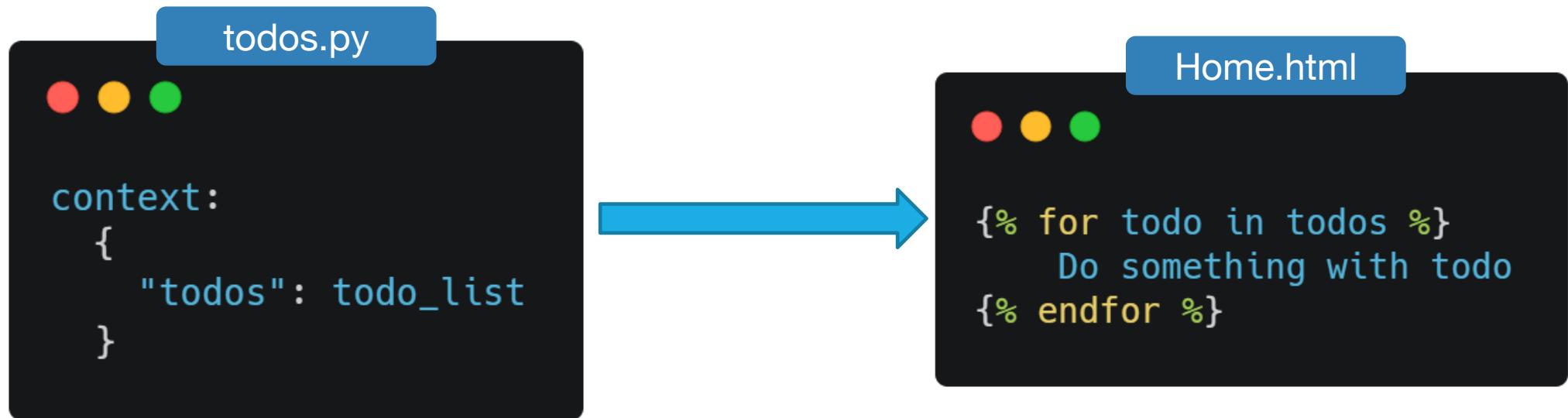
- Jinja tags allows developers to be confident while working with backend data, using tags that are similar to HTML.



```
<link rel="stylesheet" type="text/css" href="{{ url_for('static', path='/todo/css/base.css') }}>
```

WHAT ARE JINJA TEMPLATING TAGS AND SCRIPTS?

- Now imagine we have a list of *todos* that we retrieved from the database. We can pass the entire list of *todos* into the front-end and loop through each *todo* with this simple ‘for loop’ on the template.



WHAT ARE JINJA TEMPLATING TAGS AND SCRIPTS?

- We can also use Jinja templating language with *if* *else* statements. One thing that may stand out is the double brackets with *todos|length*

```
{% if todos %}  
    Displaying: {{ todos|length }} Todos  
{% else %}  
    You don't have any todos :)  
{% endif %}
```

LET'S LEARN GIT



WHAT IS GIT?



- Free and open-source distributed **version control system**
- Can handle small to large applications and projects
- Allows team members to use same files by distributed branches/environments



GIT EXAMPLE?

Version 1



```
class MathFunctions
def addition(a, b)
    return a + b
```

Simple Python Class

Addition Function

We need to add subtraction to MathFunctions

GIT EXAMPLE?

Version 2



```
class MathFunctions
```

Simple Python Class

```
def addition(a, b):  
    return a + b
```

Addition Function

```
def subtraction(a, b):  
    return a - b
```

Subtraction Function

New Function

New Version Of Code

GIT EXAMPLE?

Version 3



```
class MathFunctions
```

Simple Python Class

```
def addition(a, b):  
    return a + b
```

Addition Function

```
def subtraction(a, b):  
    return a - b
```

Subtraction Function

```
def multiplication(a, b):  
    return a * b
```

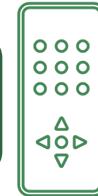
Multiple Function

GIT EXAMPLE?

Version 1

```
class MathFunctions:  
  
    def addition(a, b):  
        return a + b
```

Version Control



Commit 1

Version 2

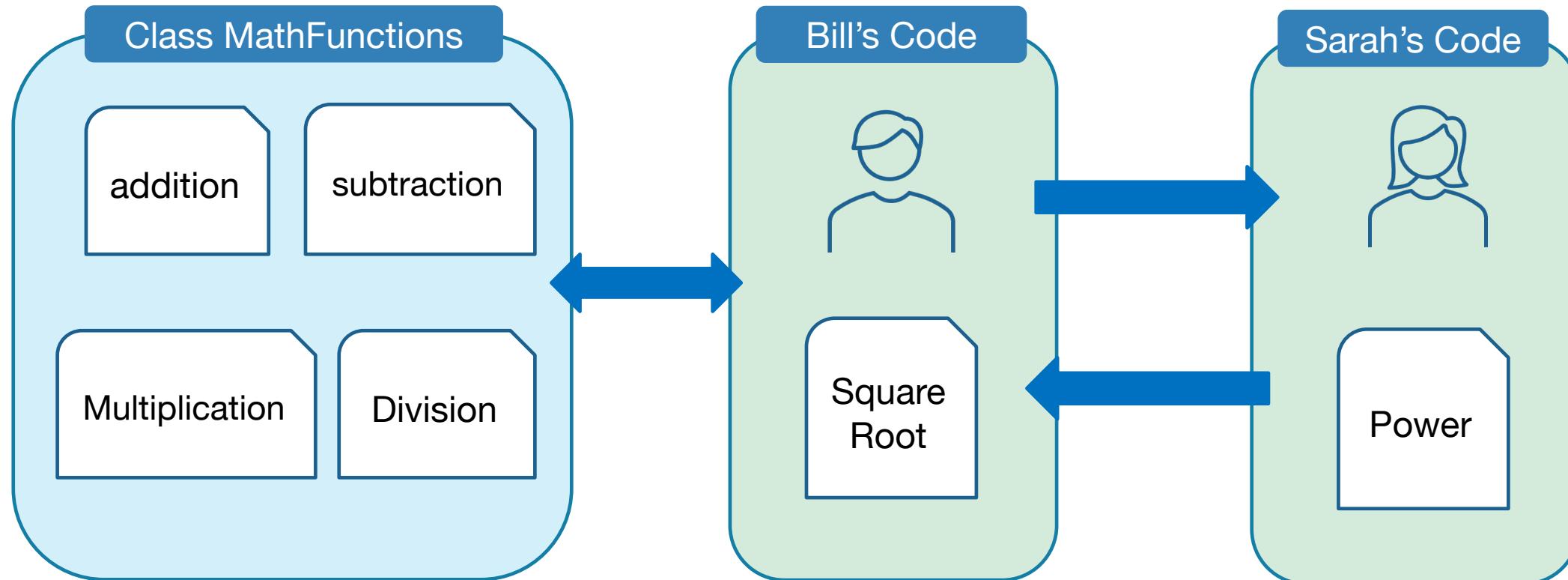
```
class MathFunctions:  
  
    def addition(a, b):  
        return a + b  
  
    def subtraction(a, b):  
        return a - b
```

Commit 2

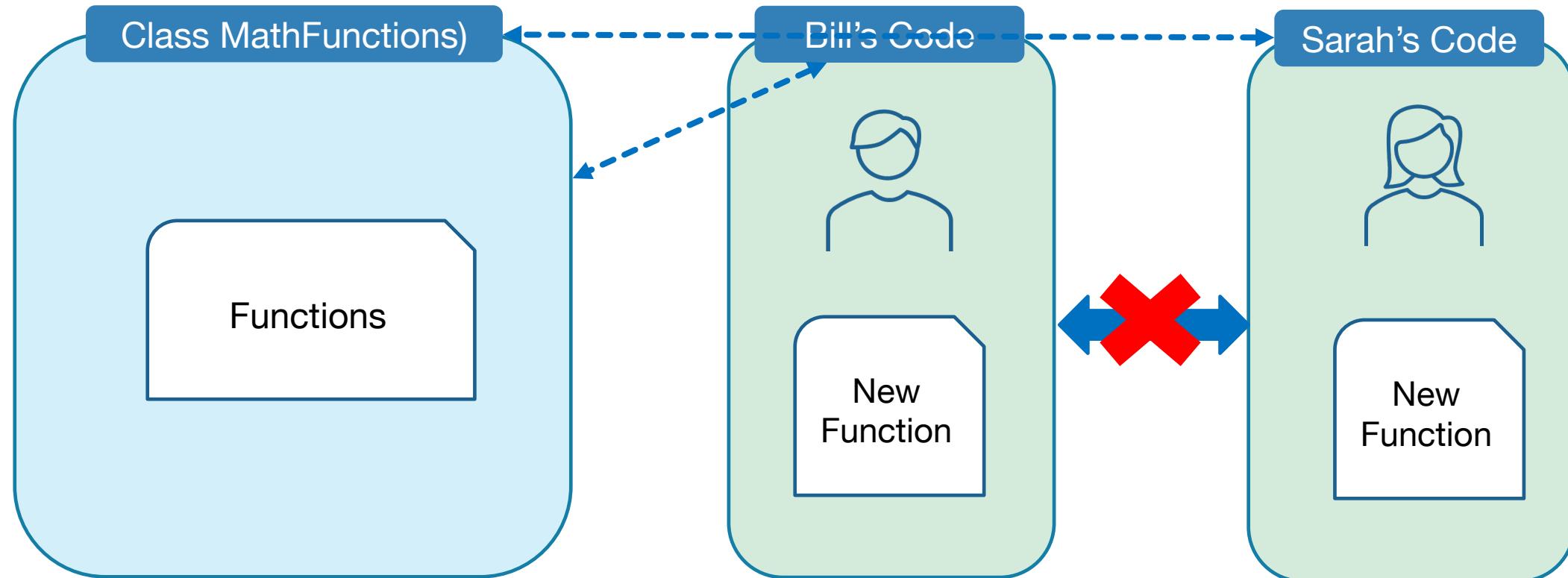
Version 3

```
class MathFunctions:  
  
    def addition(a, b):  
        return a + b  
  
    def subtraction(a, b):  
        return a - b  
  
    def multiplication(a, b):  
        return a * b
```

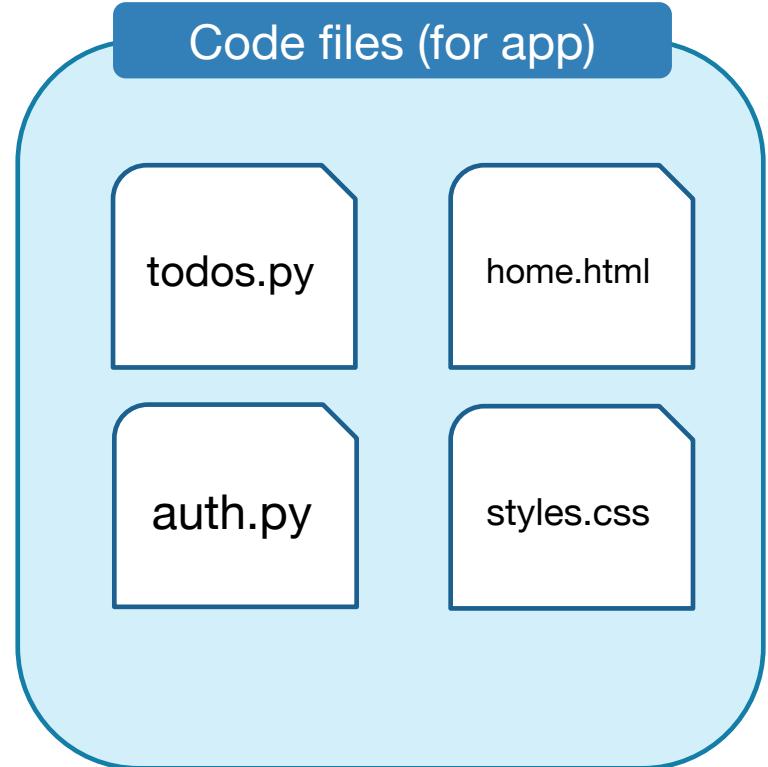
WHAT IS GIT?



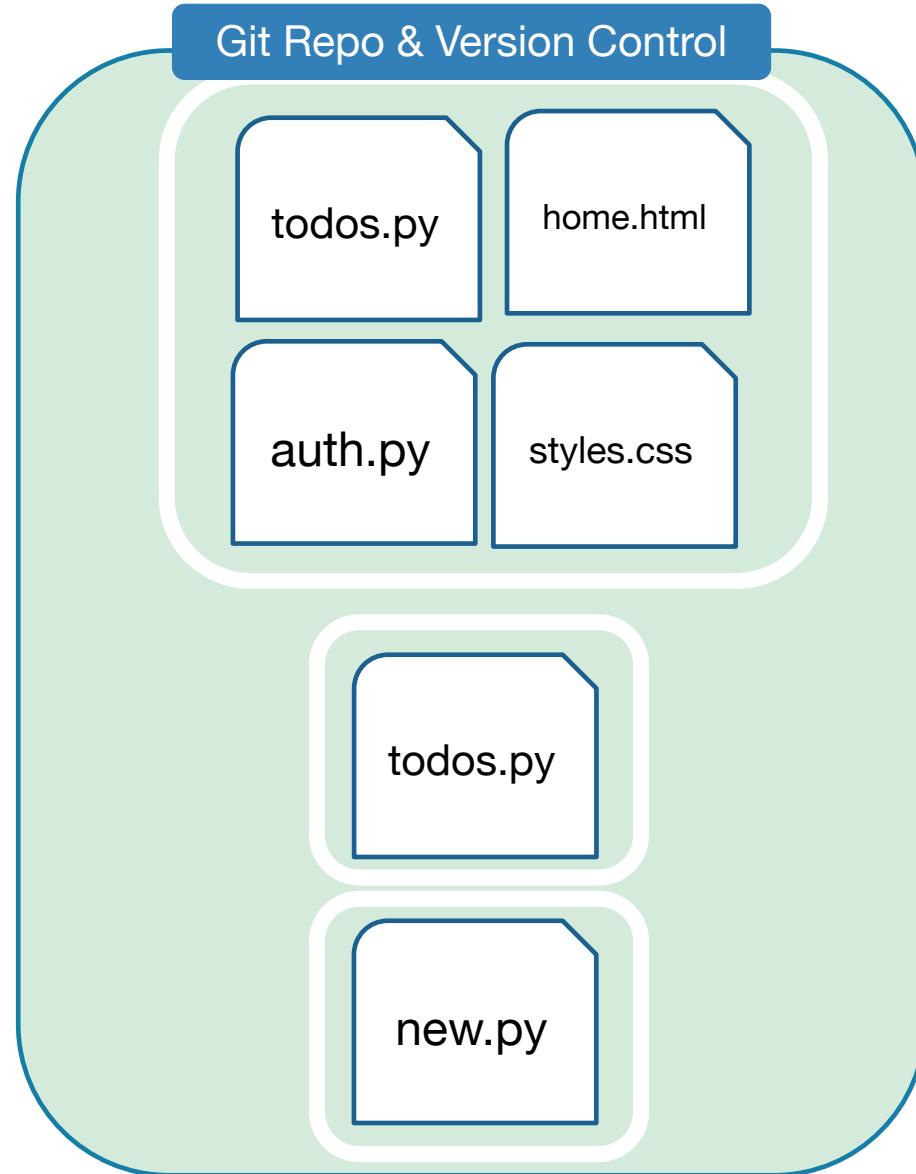
WHAT IS GIT?



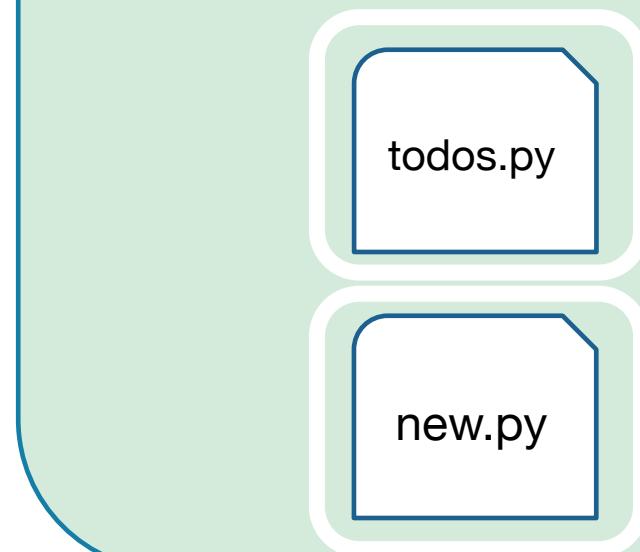
GIT?



Commit 1 →



Commit 2 →



Commit 3 →

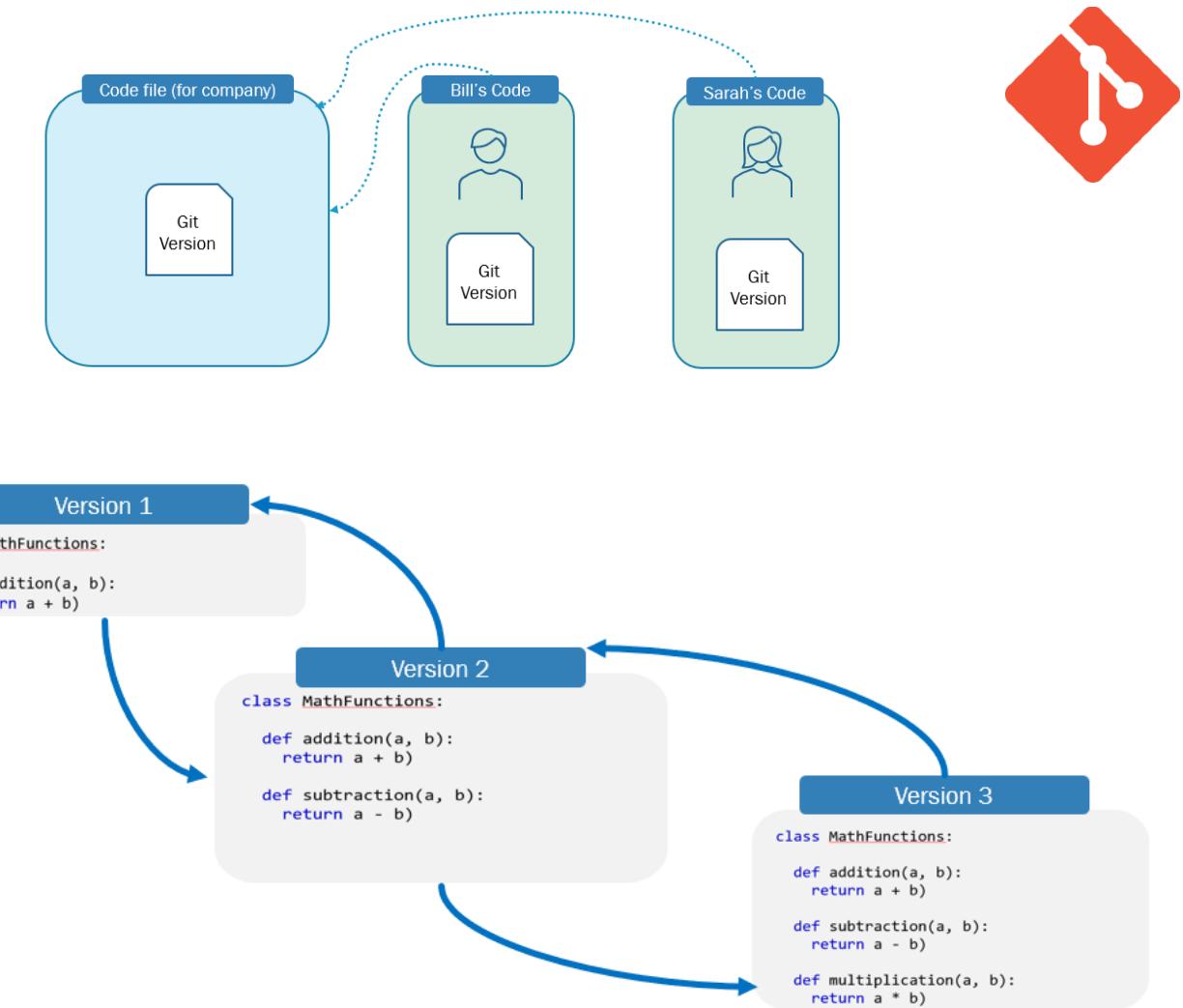


WHAT IS GIT?



git

Free and open-source distributed version control system



WHAT IS GIT?



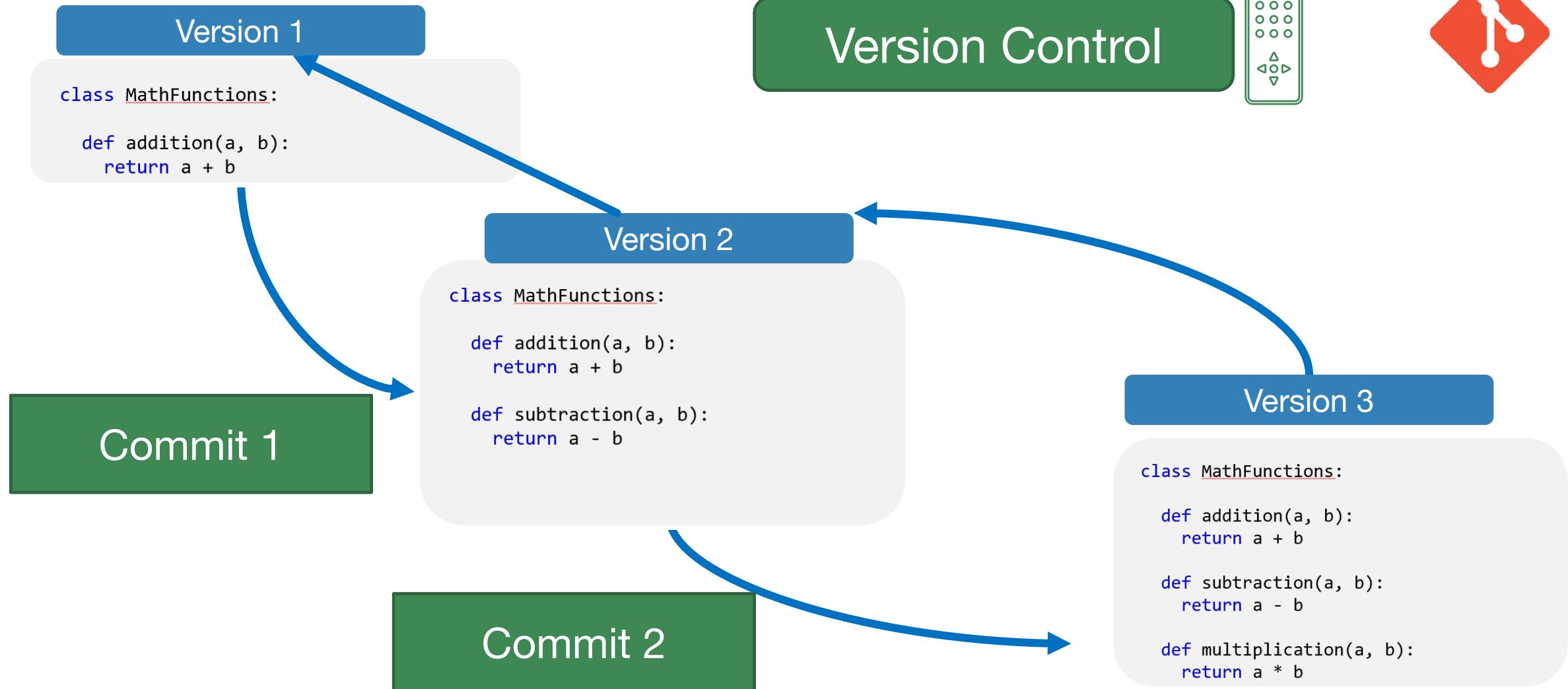
- Track Changes
- Version Control
- Allows team members to use same files without needing to sync every time



GIT BASICS



GIT EXAMPLE?



GIT EXAMPLE?



Git Command

git init

Details

Initializes a new, empty repository

GIT EXAMPLE?



Git Command

git init

git add .

Details

Initializes a new, empty repository

Adds files from a non-staged area to a staging GIT area

GIT EXAMPLE?



Git Command

git init

git add .

git commit -m "info here"

Details

Initializes a new, empty repository

Adds files from a non-staged area to a staging GIT area

Move files from a staging area to a commit

GIT EXAMPLE?



Git Command

git init

git add .

git commit -m "info here"

git checkout <commit #>

Details

Initializes a new, empty repository

Adds files from a non-staged area to a staging GIT area

Move files from a staging area to a commit

Open up previous commit

GIT EXAMPLE?



Git Command

git init

git add .

git commit -m "info here"

git checkout <commit #>

git log

Details

Initializes a new, empty repository

Adds files from a non-staged area to a staging GIT area

Move files from a staging area to a commit

Open up previous commit

Shows committed Snapshots

GIT BASICS



python-git-basics

```
class main:  
    print('Hello World')
```

GIT

git init

stashed

```
class main:  
    print('Hello World')
```

un-stashed

git add .

Commit 1 Complete

git commit -m "First
commit."

GIT BASICS

python-git-basics

```
class main:  
  
    print('Hello World')  
  
    print(2 + 2)
```

un-stashed

git add .

stashed

```
class main:  
  
    print('Hello World')  
  
    print(2 + 2)
```

Commit 2 Complete

git commit -m "Added addition functionality"



GIT BASICS



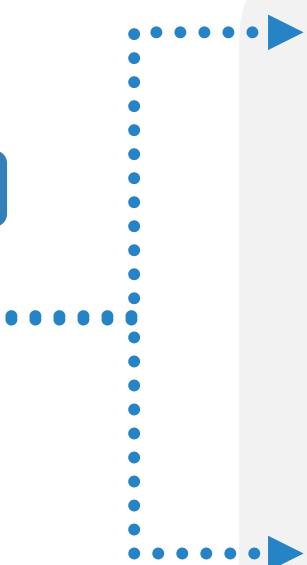
2 Different Commits

python-git-basics

git log

git checkout

aabbccddeel12233445567



commit

aabbccddeel112233445566

Author: Eric Roby

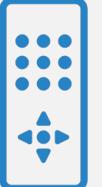
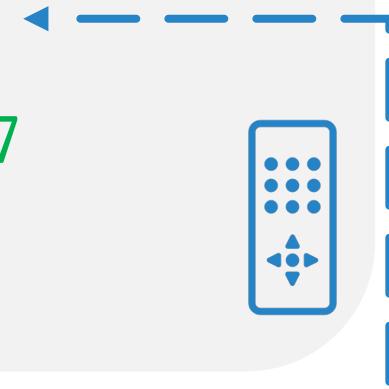
"Added Addition Function"

commit

aabbccddeel112233445567

Author: Eric Roby

"First Commit"



GIT BRANCHES



WHAT IS A GIT BRANCH?



A pointer - to take a snapshot of a change. Branches can be merged into other branches

- A pointer of data change
- Isolation of feature development
- Linear development



GIT EXAMPLE?



Git Command

git branch <branch name>

Details

Create new isolated branch

GIT EXAMPLE?



Git Command	Details
git branch <branch name>	Create new isolated branch
git checkout branch <branch name>	Change root to branch selected

GIT EXAMPLE?



Git Command	Details
git branch <branch name>	Create new isolated branch
git checkout branch <branch name>	Change root to branch selected
git switch branch <branch name>	Same as above new command as of Git (2.23)

GIT BRANCHES?



All code is currently in the ***master branch***



```
git branch multiplication-branch
```

New branch

```
git checkout multiplication-branch
```

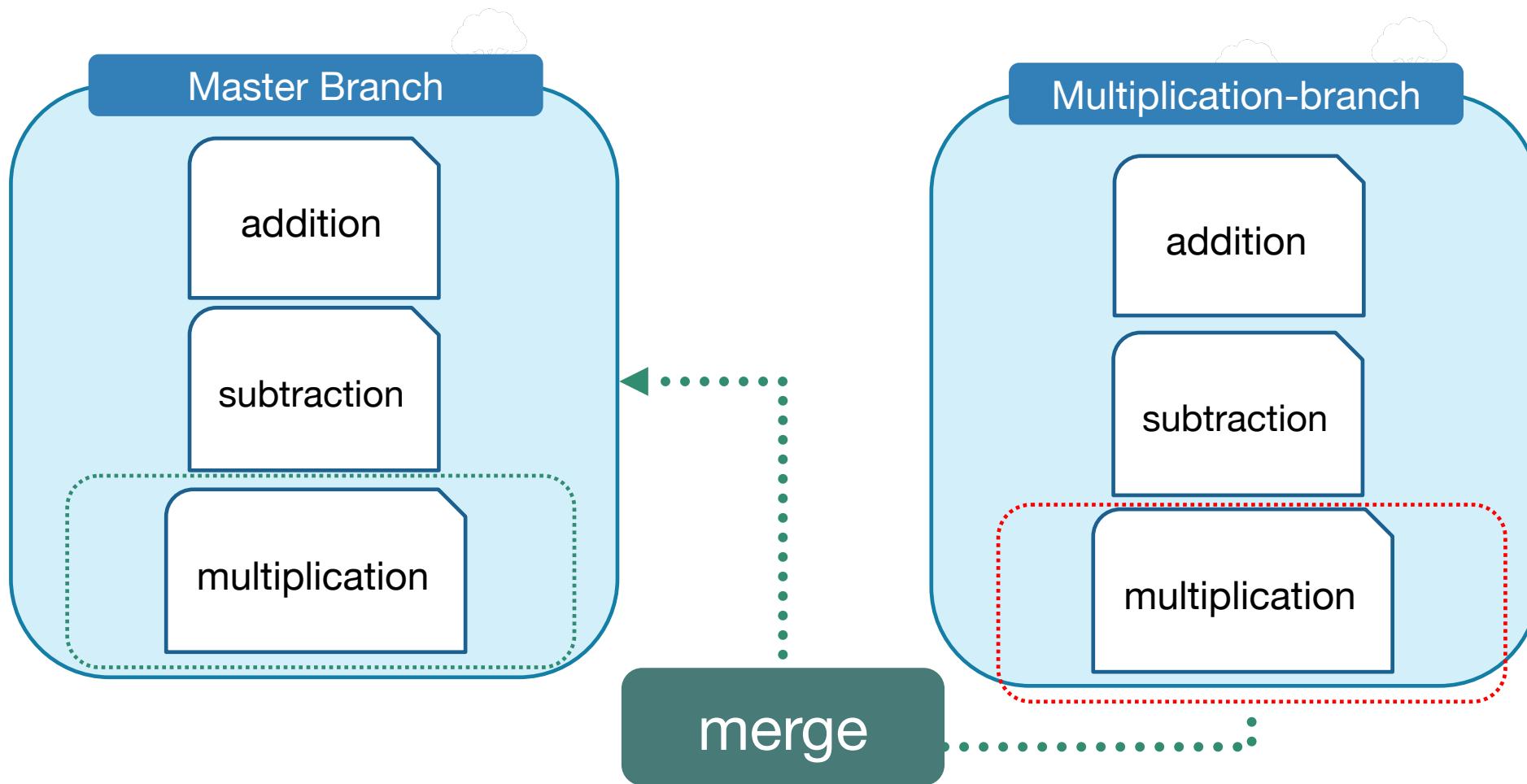
Checkout new branch

master branch



Multiplication-
branch

GIT EXAMPLE?



GIT BRANCHES

merge

master branch

git merge <commit id>

master branch

git merge multiplication-
branch

python-git-basics

```
class MathFunctions:  
  
    def addition(a, b):  
        return a + b  
  
    def subtraction(a, b):  
        return a - b
```

detached-head



multiplication-
branch

git checkout
multiplication-branch

```
class MathFunctions:  
  
    def addition(a, b):  
        return a + b  
  
    def subtraction(a, b):  
        return a - b  
  
    def multiplication(a, b):  
        return a * b
```

git commit -m "adding
multiplication function."

WHAT IS GITHUB?



WHAT IS GITHUB?

- Git Repository hosting service
- User friendly interface
- Large development platform

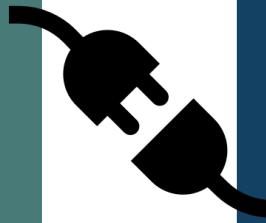


WHAT IS GITHUB?



git

Free and open-source distributed version control system



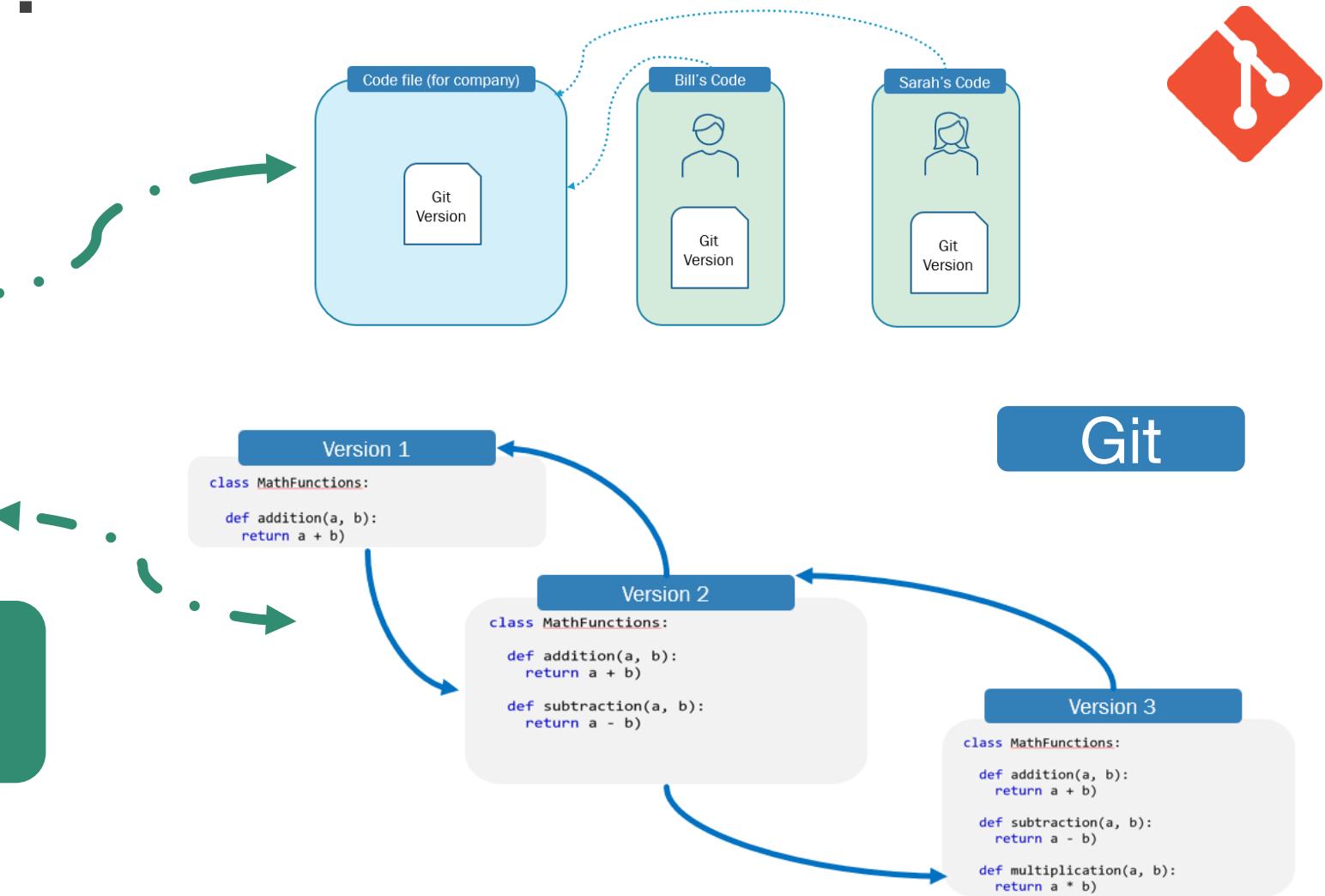
Top Git Repository hosting service available



WHAT IS GITHUB?



All Git data is saved on Cloud. Can access version control and team's merges





WHAT IS RENDER?



WHAT IS RENDER?



- Platform as a service (PaaS)
- Helps developers build, run and operate applications entirely on the cloud
- Developers can focus on coding and not have to worry about the infrastructure of their applications



RENDER PRICING?



- Pricing depends on many factors of how you want your application to perform online
- Free Trial
- Free Plan
- Businesses of all sizes use Render as their cloud PaaS provider.



WHAT IS RENDER?



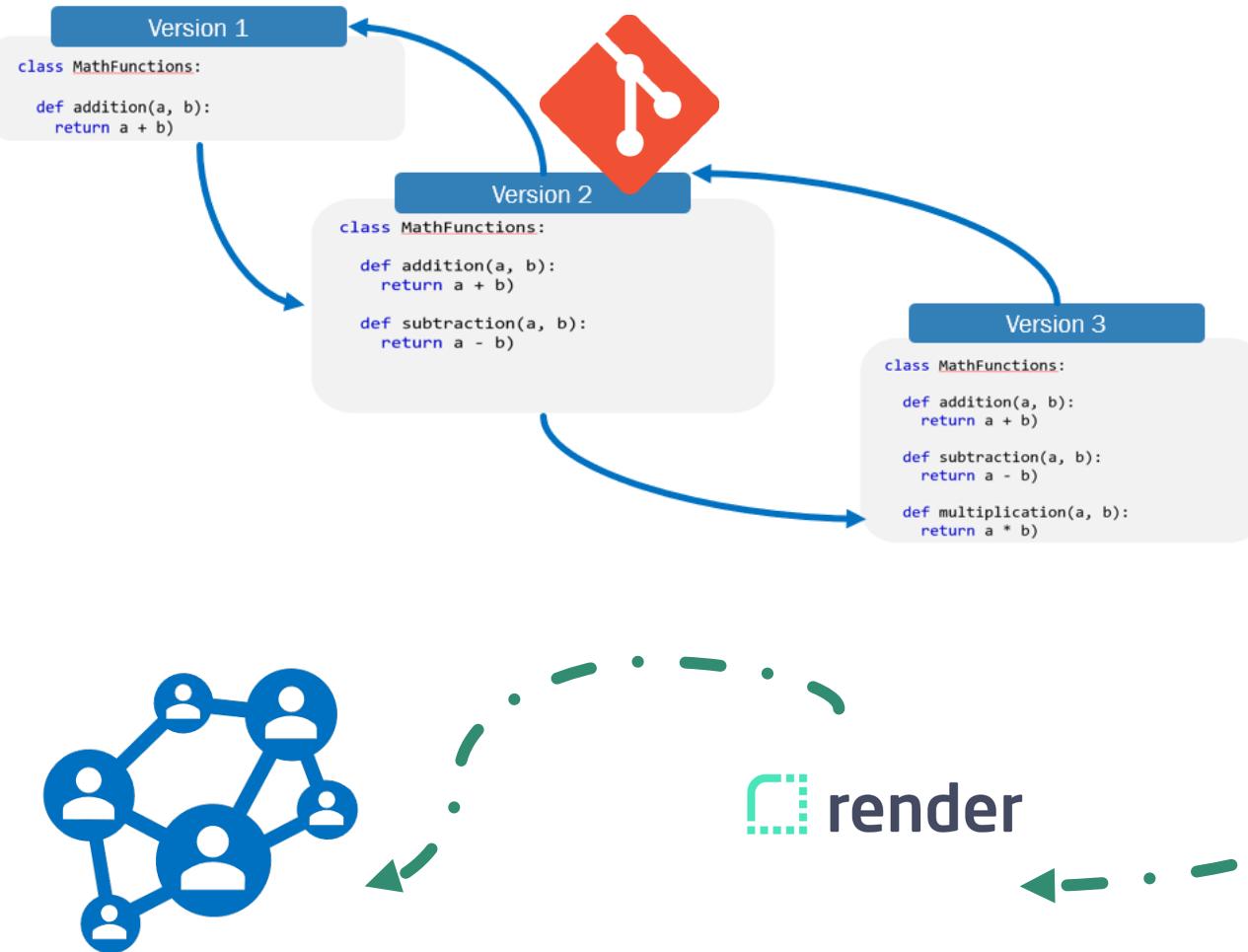
- Code deployment system
- Continuous Integration & Continuous Deployment (CI/CD)
- Load Balancing
- and more...



Focus on Code



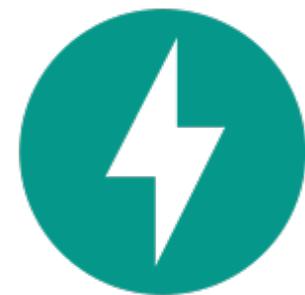
WHAT IS RENDER?



DIFFERENT PLATFORMS?



THANK YOU!!



FastAPI

CONTACT ME!

- codingwithroby@gmail.com
- Instagram: @codingwithroby

LET'S LEARN PYTHON

Assignment!

LET'S LEARN PYTHON

Variables!

LET'S LEARN PYTHON

Comments!

LET'S LEARN PYTHON

String

LET'S LEARN PYTHON

User Input!

LET'S LEARN PYTHON

Lists!

LET'S LEARN PYTHON

Sets & Tuples!

LET'S LEARN PYTHON

Boolean &

LET'S LEARN PYTHON

If Else!

LET'S LEARN PYTHON

Loops!

LET'S LEARN PYTHON

Dictionaries!

LET'S LEARN PYTHON

Functions!