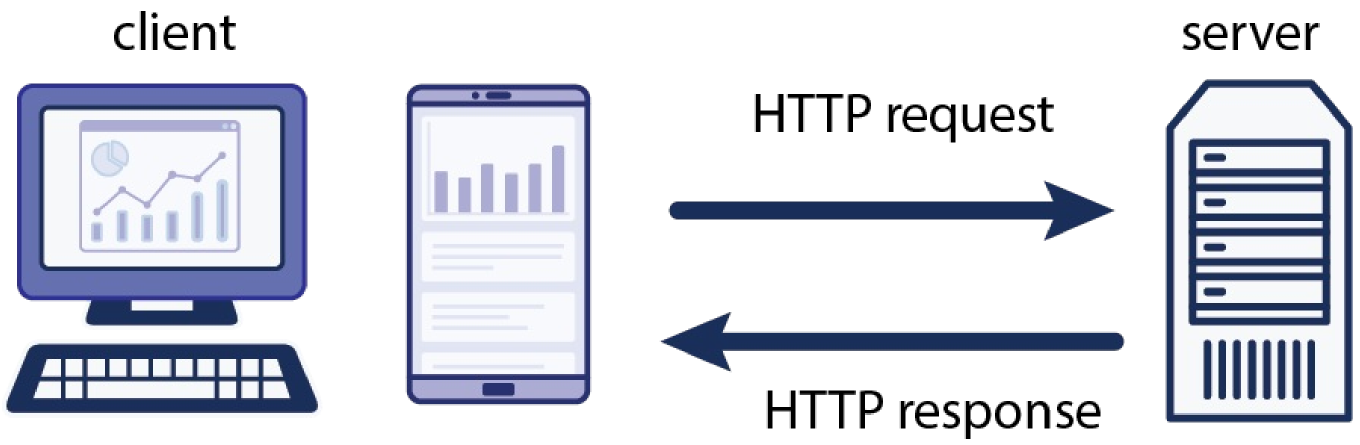
Web Apps

Class outline:

- How the web works
- Server-side Python
- CS61A projects

How the web works

Clients and servers



HTTP

A client sends an HTTP request:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

The server sends back an HTTP response:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 208
<!DOCTYPE html>
  <html>
    <head>
      <title>Example Domain</title>
    </head>
  <body>
    <h1>Example Domain</h1>
    <p>This domain is to be used for illustrative examples in documents.</p>
  </body>
</html>
```

Webpages

Webpages are made up of three languages:

- **HTML**: Contains the content and uses tags to break it into semantic chunks (headings, paragraphs, etc)
- **CSS**: Contains style rules that apply properties to elements on a page.
- **JavaScript**: Contains code that dynamically accesses and updates the page content to make it more interactive.

What does a server do?

The most basic server just serves up HTML and multimedia files from a file system.

Server-side code is also useful for anything that requires access to persistent data or needs an additional layer of security than allowed in the client.

- User authentication
- Database fetches/updates
- Caching

Server-side Python

Simple HTTP server

From the standard library, the http module can run a basic server. But it is **not** recommended for production.

Running a simple file server:

```
python -m http.server 8000
```

Demo: Simple dynamic pages

Using the http module to dynamically generate responses.

simpleserverexample.pamelafox2.repl.co/path

View the code by clicking the "Code" tab at https://replit.com/@PamelaFox2/SimpleServerExample

Based on this code.

Flask framework

An external package, Flask is a lightweight framework for server requests and responses.

Apps written in Flask:

- cs61a.org
- Khan Academy (originally)
- Reddit
- Netflix

Demo: Simple Flask website

Using the Flask framework to generate responses for each routes.

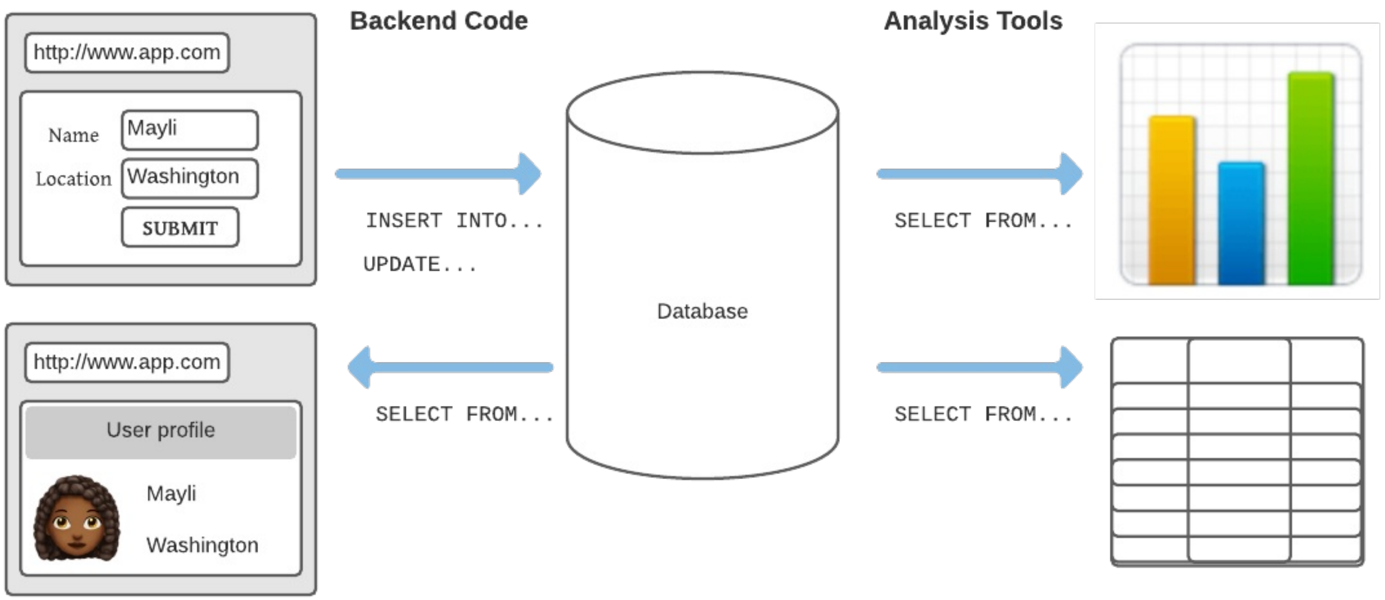
simpleflaskexample.pamelafox2.repl.co/

View the code by clicking the "Code" tab at https://replit.com/@PamelaFox2/SimpleFlaskExample

Based on this tutorial.

Webapps with SQL

SQL can be used for 1) data storage and 2) data analysis.



Demo App: Native or Not?

flask-db-example.pamelafox2.repl.co

View the code by clicking the "Code" tab at https://replit.com/@PamelaFox2/flask-db-example

Demo App: 61A Merch

flask-db-example-1.pamelafox2.repl.co/

View the code by clicking the "Code" tab at https://replit.com/@PamelaFox2/61AMerchWithTransactions

Includes a transaction:

```
BEGIN;

UPDATE products SET quantity = quantity - 1
    WHERE id = 1;

INSERT INTO orders (customer, product_id)
    VALUES ("Animesh", 1);

COMMIT;
```

Django framework

An external library, Django is a fairly heavyweight/opinionated framework for server-side code. Includes an ORM for database interaction.

Apps written in Django:

- Coursera (originally, now Scala+Play)
- Instagram
- Pinterest (originally, now Flask)
- Eventbrite

Demo: ??

CS61A projects

Common files

Hog and Cats both use the http module to run a server and share common files for setting up the server.

common_server.py

Cats code

The Cats code includes many routes for handling multiplayer games, since those require access to the database.

multiplayer.py

Hog code

The Hog code includes routes for taking the next turn. It does not store anything in a database, the browser just remembers all the turns.

hog_gui.py