OS TECH

TECH FOR NON-DEVELOPERS

GOALS

- Improve communication between technical and non-technical staff
- Give a broad but shallow understanding of a wide variety of topics
- Hands on experience as much as possible

LIMITATIONS

- This class will not teach you to code
- We will do code related activities, but they are used to understand concepts

SERVERS

TROUBLE STALKING YOUR APPETITE?
Avoid it with our NEW
BATBURGER
FLAVORS

SERVERS

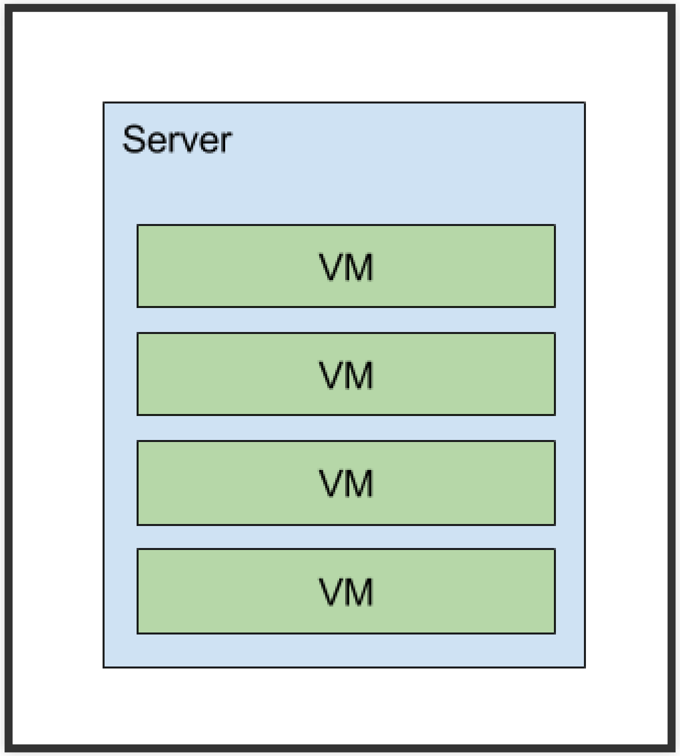
- Openstax has servers in the Rice Data Center and on Amazon AWS
- All of our servers run Ubuntu Linux as the operating system
- Most of our servers use Virtual Machines

SERVERS

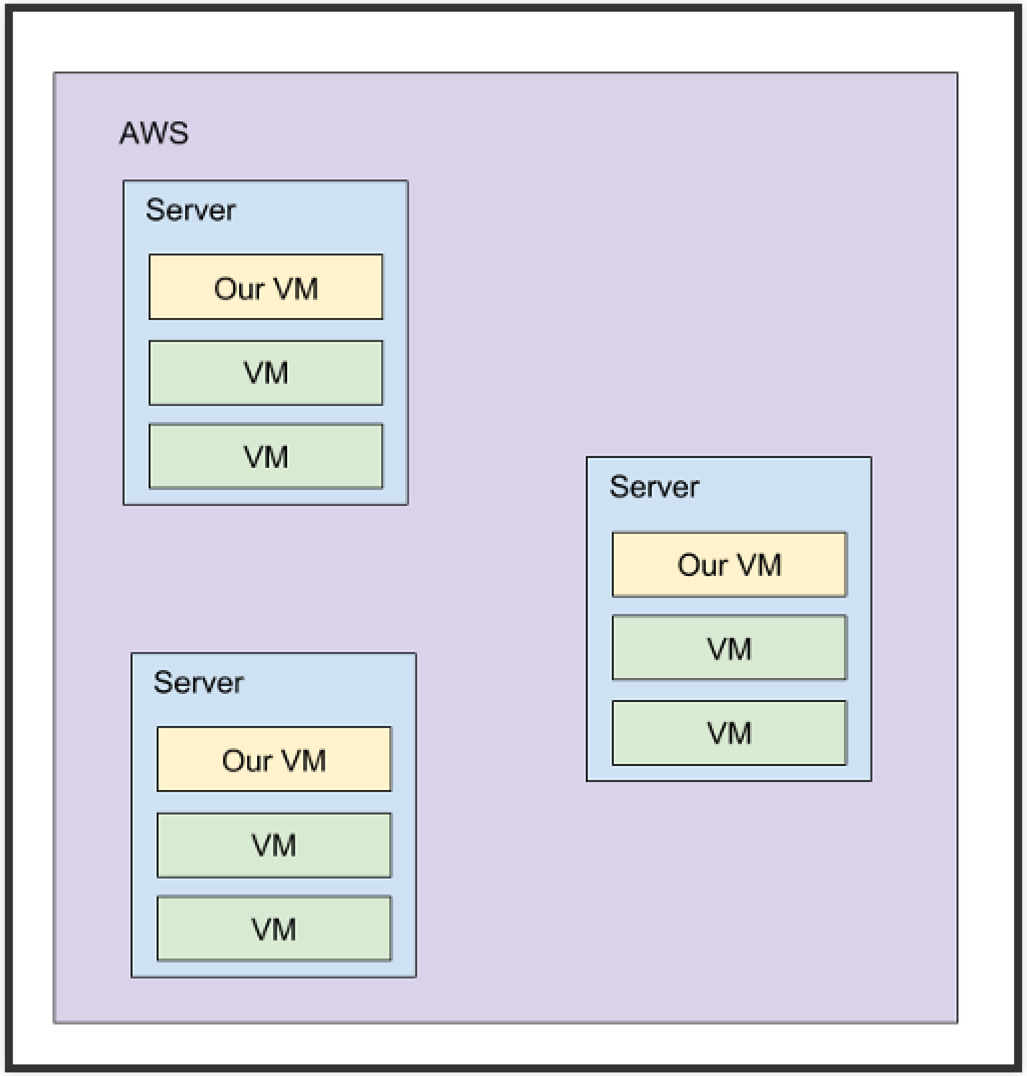
- Virtual Machines...
    - Allow big hardware to be utilized for many purposes
    - Can be easily rebuilt
    - VMs are rebuilt and updated using automated deployments
    - Deployments are automated using a tool called Ansible

SERVERS

- Virtual Machines...
  - Can have disk space and memory (RAM) adjusted as needed



VIRTUAL MACHINES IN AWS..

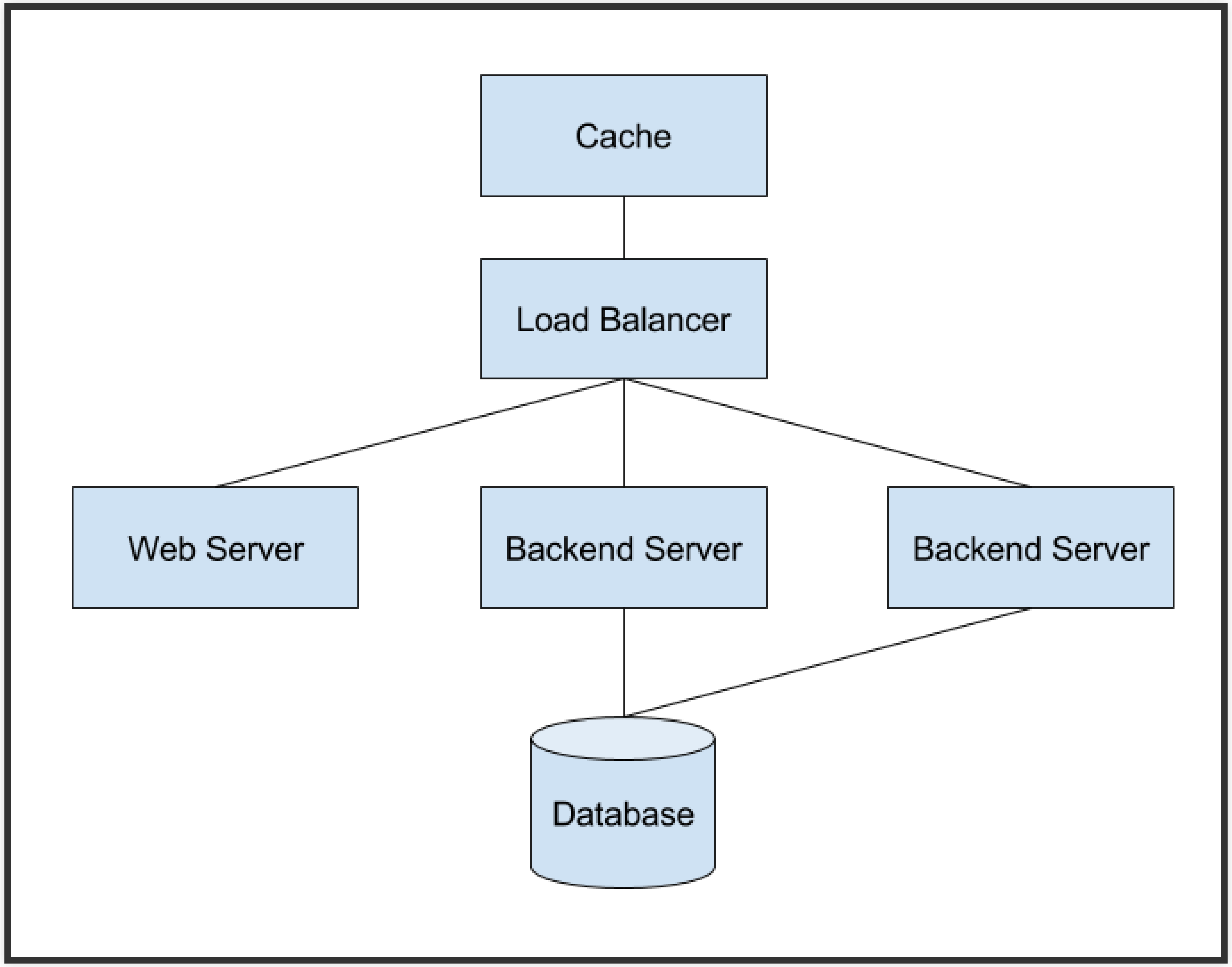


APPLICATION ARCHITECTURE - HARDWARE

APPLICATION ARCHITECTURE - HARDWARE

- Applications have similar Architectures
  - Cache (optional)
  - Load Balancer (optional)
  - Web Server
  - Backend server
  - Database server

APPLICATION ARCHITECTURE - HARDWARE


Cache
Load Balancer
Web Server
Backend Server
Backend Server
Database

APPLICATION ARCHITECTURE - HARDWARE

- Cache
  - Used to quickly return read-only content
  - Prevents expensive trip to the server and database
  - Content is retrieved once from the server and then stored in the Cache
  - A Cache is temporary storage. The Cache is emptied if the server is restarted

APPLICATION ARCHITECTURE - HARDWARE

- Load Balancer
  - A Load Balancer spreads requests across all of the Web Servers or Backend Servers
  - It is used to prevent any of the VMs from having too many requests
  - Too many requests can result in slow response times or errors

APPLICATION ARCHITECTURE - HARDWARE

- Web Server
  - A Web Server delivers the client Javascript that runs in the browser
  - It generally does not get a lot of load

APPLICATION ARCHITECTURE - HARDWARE

- Backend server
  - A Backend Server runs code that manipulates data and runs APIs
  - This code is usually in Python or Ruby
  - Might be running in what is called a WSGI server - Web Server Gateway Interface
  - Calls to this server are expensive in time

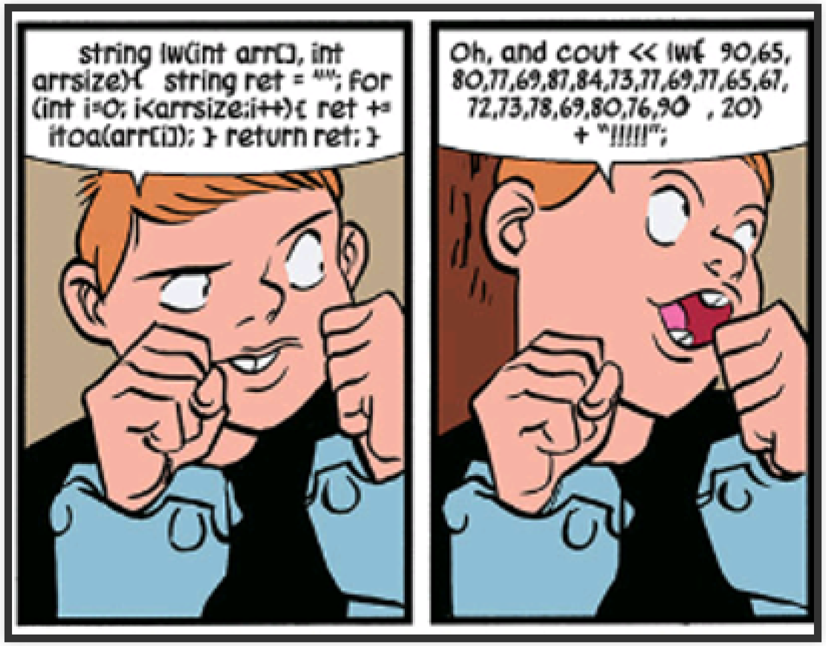
APPLICATION ARCHITECTURE - HARDWARE

- Database server
  - Runs database software such as PostgreSQL
  - Backend Server connects to the database to manipulate data
  - Actions on data are sometimes called CRUD operations
  - CRUD = Create, Read, Update and Delete

SERVER ARCHITECTURE EXERCISE

- What servers would you need if...
  - 80% of your content is read only
  - All data is stored in a database
  - Frontend is a SinglePage App
  - Each backend server can handle 200 requests at a time
  - Peak load is estimated to be 500 requests at a time

CODE



```
string lw(int arr[], int
arrsize){   string ret = ""; for
(int i=0; i<arrsize;i++){ ret +=
itoa(arr[i]); } return ret; }
```

```
Oh, and cout << lw{  90,65,
80,77,69,87,84,73,77,69,77,65,67,
72,73,78,69,80,76,90  , 20)
+ "!!!!!";
```

APPLICATION ARCHITECTURE - CODE

- Model View Controller - MVC
  - Model - holds data. Can have logic related to data (sorting, filtering, etc.)
  - View - display of model. Little if any logic
  - Controller - handles requests and calls business logic

APPLICATION ARCHITECTURE - CODE

- Model View Controller - MVC
  - Why MVC?
  - Separation of Concerns
  - Testability
  - Less bugs
  - Easier to maintain

APPLICATION ARCHITECTURE - CODE

- Single Page App
  - Frontend is a structure and what is inside the structure changes
  - Structure is rebuilt on the frontend rather than the backend

APPLICATION ARCHITECTURE - CODE

- Single Page App
  - Model - represents the data for the page.
  - View - the display of the data. Should only have code used for display.
  - Controller - orchestrates calls between FE and BE

APPLICATION ARCHITECTURE - CODE

- Database
  - The data is stored in tables and accessed using a language called SQL
  - SQL is also called "See-Qwill"

APPLICATION ARCHITECTURE - CODE

- Database
  - Example SQL
    - select * from users
    - select * from users where city = 'Houston'
    - select first_name, last_name from users where state = 'TX'
    - select users.last_name, products.name from users, products where users.id = products.user_id

APPLICATION ARCHITECTURE - CODE

- SQL Writing Exercise
  - Query to select all users from Texas and Lousiana. Should display their first and last name along with the state
    - Ed Woodward LA
    - Stan Lee TX
    - Jack Kirby TX
    - Steve Ditko LA

USER TABLE

first_name	last_name	state
Ed	Woodward	LA
Stan	Lee	TX
Jack	Kirby	TX
Peter	Parker	NY

APPLICATION ARCHITECTURE - API

- API - Application Programming Interface
- There are different types of APIs
- We use REST APIs
- REST APIs are APIs available over http

APPLICATION ARCHITECTURE - API

- REST APIs are accessed via URLs
- EXAMPLE: https://archive.cnx.org/contents/031da8d3-b525-429c-80cf-6c8ed997733a.json
- REST APIs return json
- Json is a format of name-value pairs

APPLICATION ARCHITECTURE - JSON

```
contents: [
                {
                shortId: "4SMp5I1s@2",
                id: "e12329e4-8d6c-49cf-aa45-6a05b26ebcba@2",
                title: "Introduction to One-Dimensional Kinematic
                },
                {
                shortId: "_H3EiPTs@4",
                id: "fc7dc488-f4ec-498c-b261-3b75eaa70aaa@4",
                title: "Displacement"
                },
                {
                shortId: "gj9KHj0Q@2",
                id: "823f4a1e-3d10-44b6-a22e-8608f72c6eca@2",
                title: "Vectors, Scalars, and Coordinate Systems"
                },
```

JSON EXAMPLES

- https://openstax.org/api/pages/technology/
- https://openstax.org/api/pages/accessibility/

APPLICATION STACK

APPLICATION STACK - CNX

- Backbone - Javascript framework
- Python backend - no framework
- Postgresql Database

APPLICATION STACK - LEGACY CNX

- Plone/Zope - no Javascript framework
- Python
- Postgresql Database

APPLICATION STACK - OPENSTAX.ORG

- Superb - Javascript framework
- Django with Wagtail CMS - Python
- Postgresql Database

APPLICATION STACK - TUTOR

- React - Javascript framework
- Ruby on Rails - CMS/Framework
- Ruby
- Postgresql Database

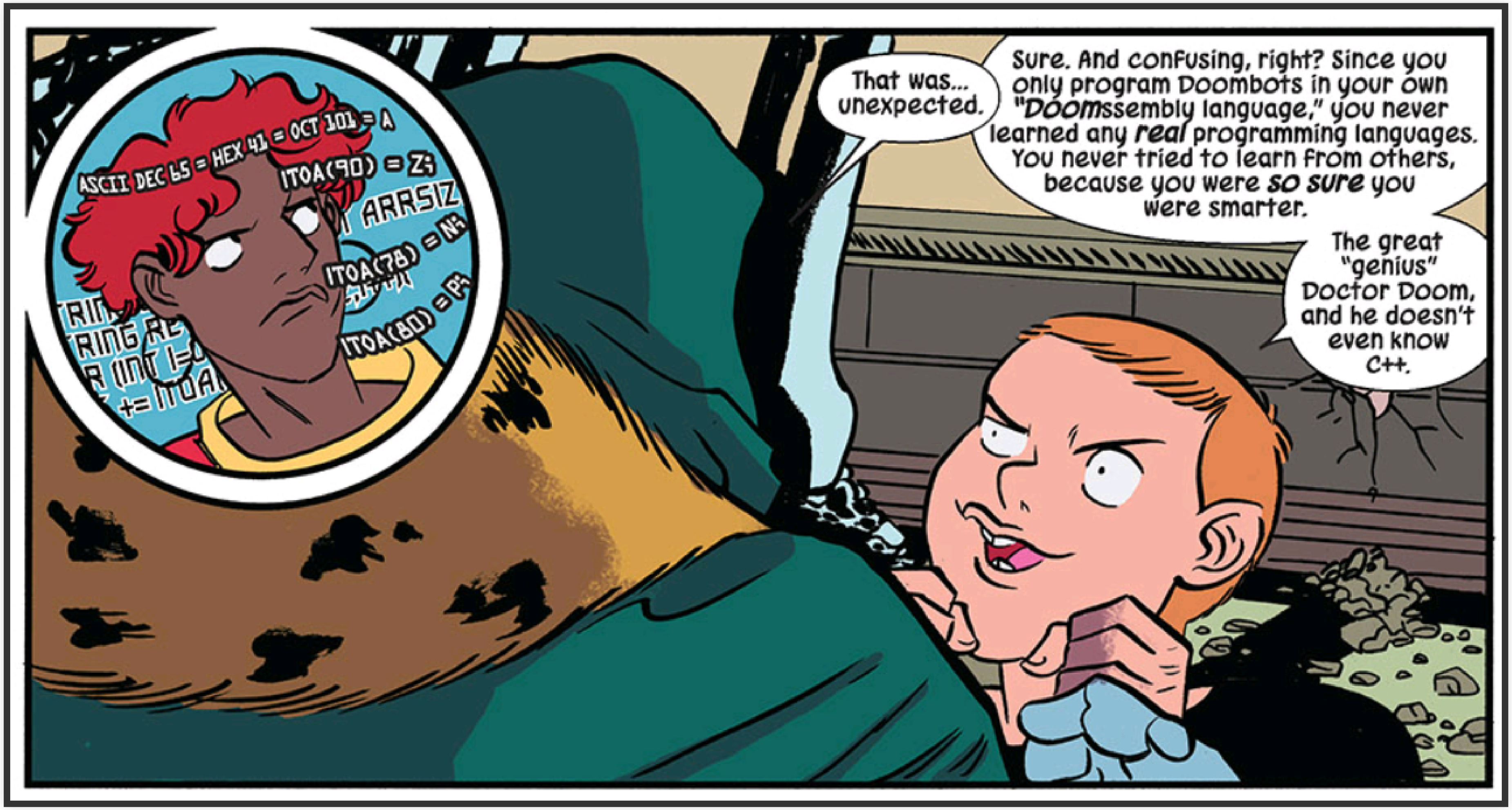
APPLICATION STACK - PAYMENTS

- React - Javascript framework
- Django CMS
- Python
- Postgresql Database

APPLICATION STACK - ACCOUNTS

- Ruby on Rails - CMS/Framework
- Ruby
- Postgresql Database

CODE BASICS


ASCII DEC 65 = HEX 41 = OCT 101 = A
ITOA(90) = Z;
ARRSIZ
ITOA(78) = N;
ITOA(80) = P;
That was... unexpected.
Sure. And confusing, right? Since you only program Doombots in your own "Doomssembly language," you never learned any real programming languages. You never tried to learn from others, because you were so sure you were smarter.
The great "genius" Doctor Doom, and he doesn't even know C++.

CODE BASICS

- Classes
- Methods/Functions
- Why use a Framework?

CODE BASICS - CLASS

- Classes represent an object in code
- Classic example is a person

CODE BASICS - PYTHON CLASS

```python
class Person():
    def __init__(firstname, lastname, city, state)
        self.first_name = firstname
        self.last_name = lastname
        self.city = city
        self.state = state

    def get_name():
        return self.first_name + ' ' + self.last_name

    def get_address():
        return self.city + ' ' + self.state
```

CODE BASICS - WHY CLASSES?

- Makes code more understandable
- Makes code reusable
- Makes code easier to maintain

CODE BASICS - METHODS/FUNCTIONS

- Methods and Functions are the same thing
- Called different things in different programming languages

CODE BASICS - METHODS/FUNCTIONS

- Methods are a self contained block of code that does one thing
- They need to do one thing so there are no unexpected side effects

CODE BASICS - PYTHON FUNCTION

```
def get_name():
    return self.first_name + ' ' + self.last_name

def get_address():
    return self.city + ' ' + self.state
```

CODE BASICS - FRAMEWORKS

CODE BASICS - FRAMEWORKS

- Why use a framework?
- Allows for faster development

CODE BASICS - FRAMEWORKS

- Django demo

CODE BASICS - FRAMEWORKS

- Frameworks we use
  - React
  - Backbone
  - Rails or Ruby on Rails
  - Django
  - Wagtail

CODE BASICS - PYTHON

- Loosely Typed
- Not compiled, but interpreted
- CNX used 2.7
- Legacy uses 2.4
- OS Web and Payments use 3.5

CODE BASICS - PYTHON EXERCISE

- Open terminal
- Type 'python --version'

CODE BASICS - PYTHON EXERCISE

- Type 'python' in terminal
- add 2 numbers together: 2 + 2 and hit Enter
- Do other math
    - Multiplication - *
    - Division - /
    - Mod - %

CODE BASICS - PYTHON EXERCISE

```
>>> the_world_is_flat = 1
>>> if the_world_is_flat:
...     print 'Be careful not to fall off!'
```

CODE BASICS - PYTHON EXERCISE RESULTS

```
>>> the_world_is_flat = 1
>>> if the_world_is_flat:
...     print 'Be careful not to fall off!'
...
Be careful not to fall off!
```

CODE BASICS - JAVASCRIPT

- Originally designed to run in the browser
- Now sometimes used on the server
- Javascript is not Java

CODE BASICS - JAVASCRIPT

```javascript
function assignColors(team) {
    let i = 0;

    for (const member of team) {
        member.bgColor = colorCombos[i][0];
        member.textColor = colorCombos[i][1];
        i++;

        if (Math.random() > 0.9) {
            i++;
        }

        i %= colorCombos.length;
    }
}
```

CODE BASICS - GIT

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

CODE BASICS - GIT

- Git is a distributed version control system
- Version control protects the code
- Prevents accidental loss of code

CODE BASICS - GIT

- Each check in of code is versioned
- Allows development of new featues while still allowing access to current code in production

CODE BASICS - GITHUB

- Github provides services on top of Git including
  - Issue tracking
  - Viewing code via the web
  - Pull Requests

CODE BASICS - GIT/GITHUB

- Common Git/Github Terminology
  - Repository - a logical grouping of code or a project
  - Clone - check out a Repository
  - Pull - get recent code from Github
  - Branch - a pointer with a name to a snapshot of the code

CODE BASICS - GIT/GITHUB

- Common Git/Github Terminology
  - Branch - same branch can be local and in Github
  - Commit - add changes to your branch
  - Rebase - copy recent changes into your branch
  - Pull Requests - requests to add code back to the project

CODE BASICS - GIT/GITHUB

- Clone a Repository
  - Type
  - git clone https://github.com/edwoodward/os-tech-class.git

GIT BASICS - BRANCHING

- list branches: git branch
- create a branch: git branch [branch name]
- checkout branch: git checkout [branch name]

GIT BASICS - CHANGES

- Edit one of the files
- See changes: git status
- Add changes to git: git add [file name]
- Commit Changes: git commit -m 'your comment here'

GITHUB BASICS - PULL REQUEST

- push to github: git push origin [branch name]
- Go to https://github.com/edwoodward/os-tech-class
- Select Pull Request tab
- Create PR to merge your branch into master

GITHUB BASICS - PULL REQUEST

- Look at your branch in Github
- SHA is on right side
- SHA is a hash that represents the version of the code

RELEASE PROCESS - SERVERS

- Dev - used for initial testing of code
- QA - formal testing by QA team
- Staging - replica of production for final testing
- Production - user facing application

RELEASE PROCESS

- Automated tests are run on every PR
- Some dev teams verify code on dev server
- QA team tests new features on QA
- QA team does regression testing on Staging
- Code is released to Production

QUESTIONS?