INTRODUCTION TO WEB DEVELOPMENT WITH PYTHON

Created by Taylor Barnett / @taylor_atx

WHO AM I?

WHO IS THIS TALK FOR?

If you have never created a web application

Mobile developers wanting to expand their projects

I'M GOING TO STOP A LOT IN THIS TALK.

SETUP

If you don't know Python, that's cool.

SETUP

Needed: Python, pip, virtualenv, Flask

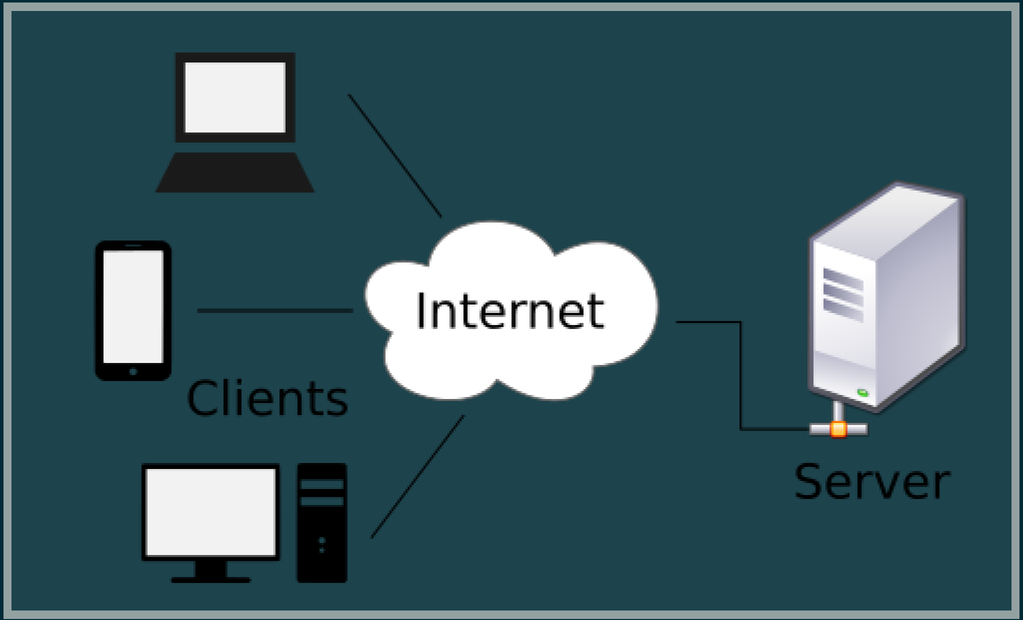
QUESTIONS AND HELP ON SETUP?

CLIENT-SERVER MODEL SIMPLIFIED

Server takes functions that requests from clients (your web browser)

Returns web content to be displayed by the client

CLIENT-SERVER MODEL SIMPLIFIED



"Client-server-model" by Gnome-fs-client.svg: David VignoniGnome-fs-server.svg: David Vignoniderivative work: Calimo (talk) - Gnome-fs-client.svgGnome-fs-server.svg. Licensed under LGPL via Wikimedia Commons.

RESTAURANT MODEL

Customer (web browser) gets their meal (web page) by telling the waiter (Flask server)

Waiter (Flask server) takes the order to a cook (back end functions)

Waiter (Flask server) returns the cooked meal to the customer (web browser)

Can serve static pages (meals off the menu)

Can also serve dynamic pages (made to order) that are generated each time you order

WHAT IS FLASK?

Flask is a framework: a collection of packages which allow developers to write web applications without having to handle low-level details such as protocols, sockets, process/thread management

Flask is a micro framework: a simple core but extensible

Does not include a database abstraction layer, like Django

Instead supports extensions to add functionality

Such as: Database integration, form validation, upload handling, various open authentication technologies, and more

Flask Extensions: http://flask.pocoo.org/extensions/

WHAT'S THE DIRECTORY STRUCTURE?

```
ProjectDirectory/
├── app.py
├── requirements.txt
├── static/
│   ├── css/
│   ├── img/
│   └── js/
└── templates/
```

`ProjectDirectory/` - Everything for your app goes in this folder, name this to the name of your app

Let's call our's `helloflask`

`app.py` - All of the Python code and server logic gets written in this file `requirements.txt` - A list of all of the dependencies for your project `templates/` - This folder holds all your Flask templates. Our HTML files will go here. There are special features offered by Flask that make templates different than basic HTML files for forms and buttons `static/` - This folder holds all your static files.Static files include: `css/` - CSS files, which style our app `js/` - Javascript files, which allow interactive web content `img/` - Image files

WHAT IS A VIRTUAL ENVIRONMENT?

Inside of your `helloflask/` directory

```
virtualenv venv --distribute
source venv/bin/activate
pip install flask
```

LET'S EDIT APP.PY:

You have to import the Flask class from the flask module:

```
from flask import Flask
```

Construct the Flask app variable, you will use this variable to access information about the server

We pass `__name__` into the `Flask()` function so that your flask app is associated with the directory structure

```
app = Flask(__name__)
```

Now we are applying the `@app.route("/")` decorator to a function called `hello()`, by doing this we are making a route

The `route()` decorator binds the URL http://www.websitename.com/ to this function of `hello()`

Functions with the `route()` decorator can return things (HTML, JSON, etc.) to the client

```
@app.route("/")
def hello():
    return "Hello World!"
```

Finally we are going to call the code below when the file app.py is executed, once the function runs the server will start accepting requests from the client

```
if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

ANYONE NEED HELP?

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

LET'S RUN IT!

While in your project directory, type this into the terminal:

```
python app.py
```

View it in your browser at localhost:5000

ANYONE NEED HELP?

WHAT CAN YOU DO NEXT?

Use templates and Jinja to make dynamic web pages

Go build a blog with all kinds of features

Awesome tutorial for it:
http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

Implement an API with the requests library

Suggestion: Twilio API (great docs and uses Flask too!)

Build out your first API (and use it in another web/mobile app): http://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

Deploy your app via Heroku:
https://devcenter.heroku.com/articles/getting-started-with-python

FEEL FREE TO TEACH OUT TO ME!

taylormbarnett@utexas.edu

I don't know a lot of the answers, but I can help you find them.