

Welcome to Web Architecture!



WEB 1.1



Welcome!



Check-In (5 minutes)

In a group of 3:

- **Share** your name, pronouns, & where you're currently living
- **Answer** the following questions:
 - What is **one great study habit** you already have that you want to **continue** doing in Term 2?
 - What is **one new study habit** that you want to **start** doing in Term 2?

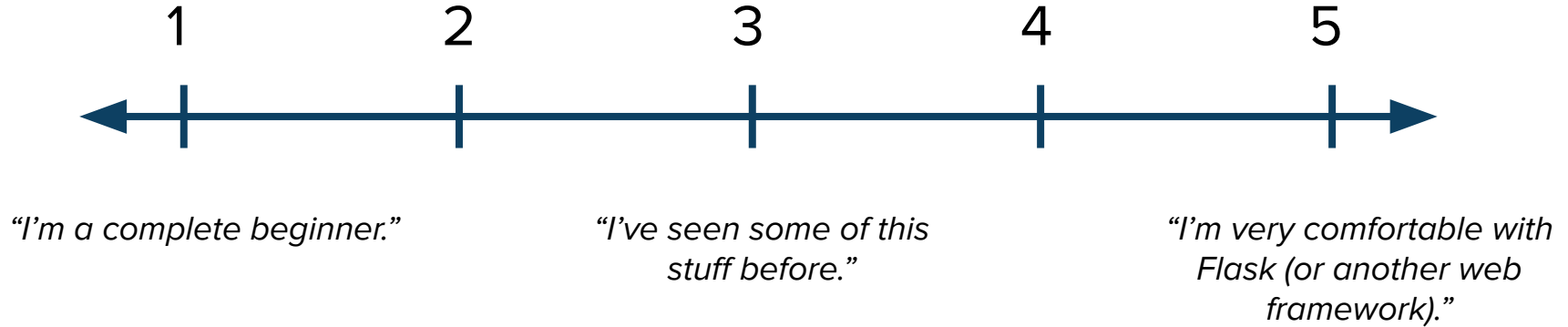
- Check-In
- Learning Outcomes
- Routes & the Request/Response Cycle
- **BREAK**
- Hello, World!
- Deconstructing Hello World
- (Time permitting) Route Variables
- Class Info
- Wrap-Up

By the end of today, you should be able to...

1. **Describe** how the Request/Response Cycle works to send you data.
2. **Identify** the request & the response for a given Internet route.
3. **Use** the Flask framework to write a route function and run it in your browser.

Routes & the Request/Response Cycle

How much do you know about web frameworks?



In today's class, we'll be answering the following question:

When I type in a web address, and press “Enter”, what happens? E.g. How does the web content (e.g. images, text, etc) get to my browser?

Warm-Up Question (5 minutes)

Journal Exercise (3 min): How does data get to my browser when I visit a website? Write down everything you know.

Breakout (3 min): Share your answers in a group of 3.

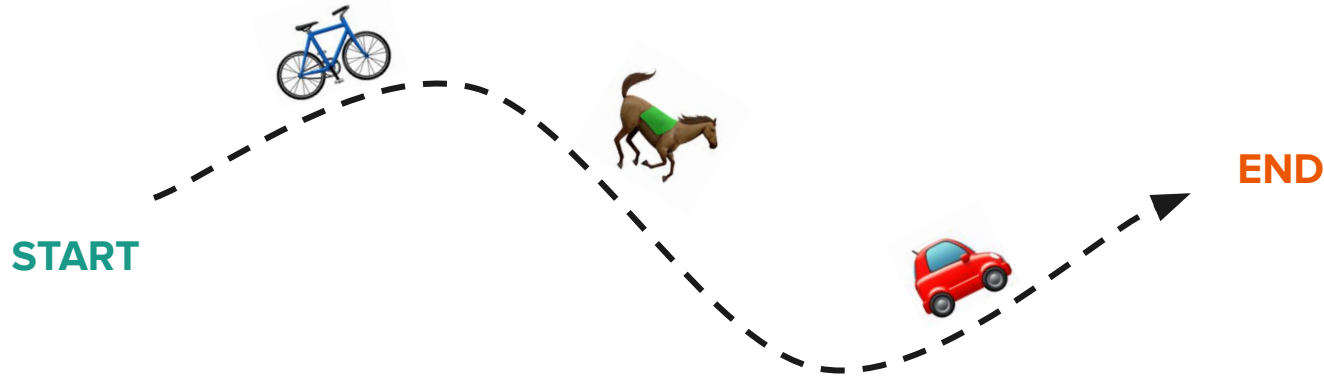
🙋 / **Raise your hand** 🙋 / if you've ever been on a **road trip** 🚗 / !

Answer in the chat:

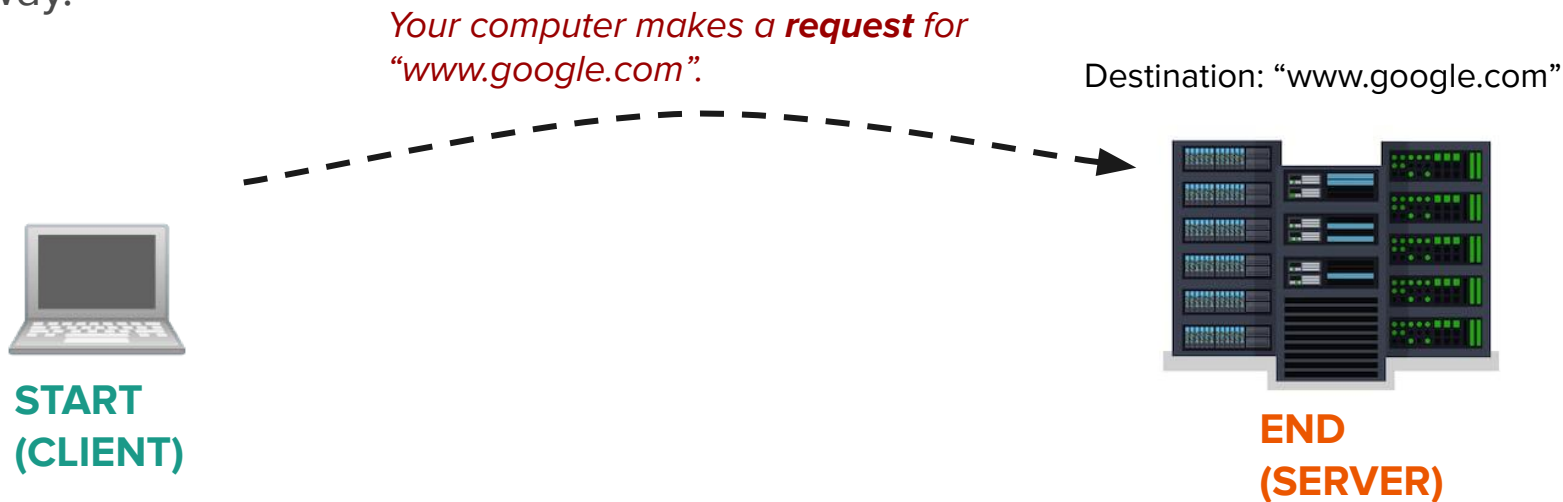
1. **Where** did you go?
2. **How long** did it take to get there?

Example: I took a road trip to Los Angeles and it took 6 hours.

A **route** is a path from a source to a destination. For example, if I want to drive from San Francisco to Los Angeles, I can take the **I-5 route**. No matter my method of travel, I'll get there eventually.



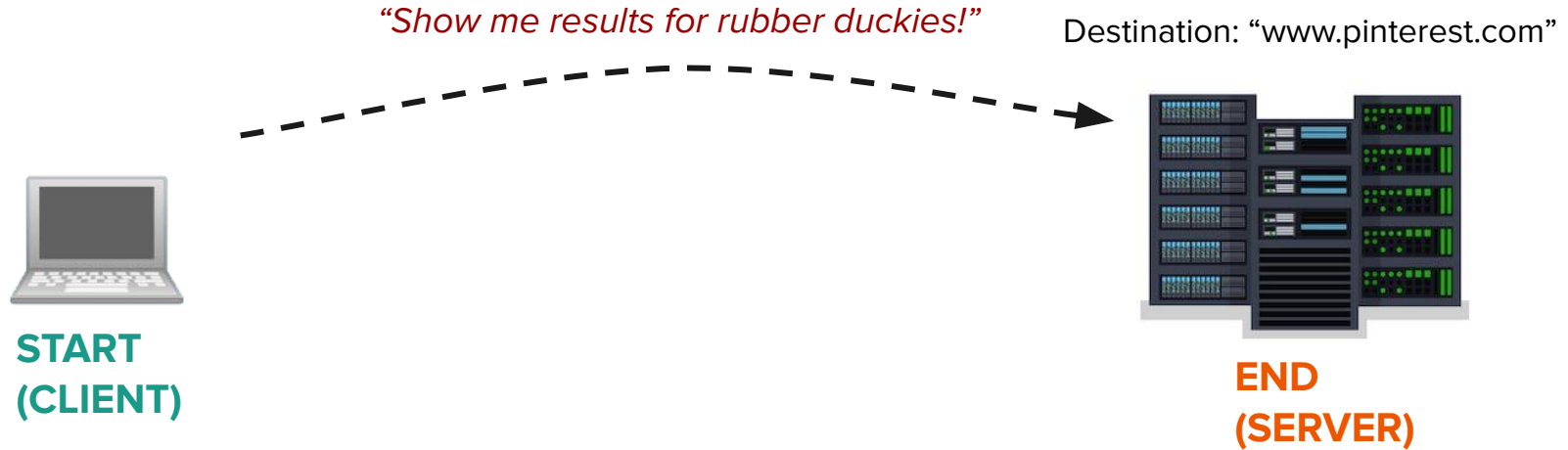
Likewise, an **Internet route** is a path from your computer to a destination **server**. These servers usually live in a **datacenter** which could be located far away.



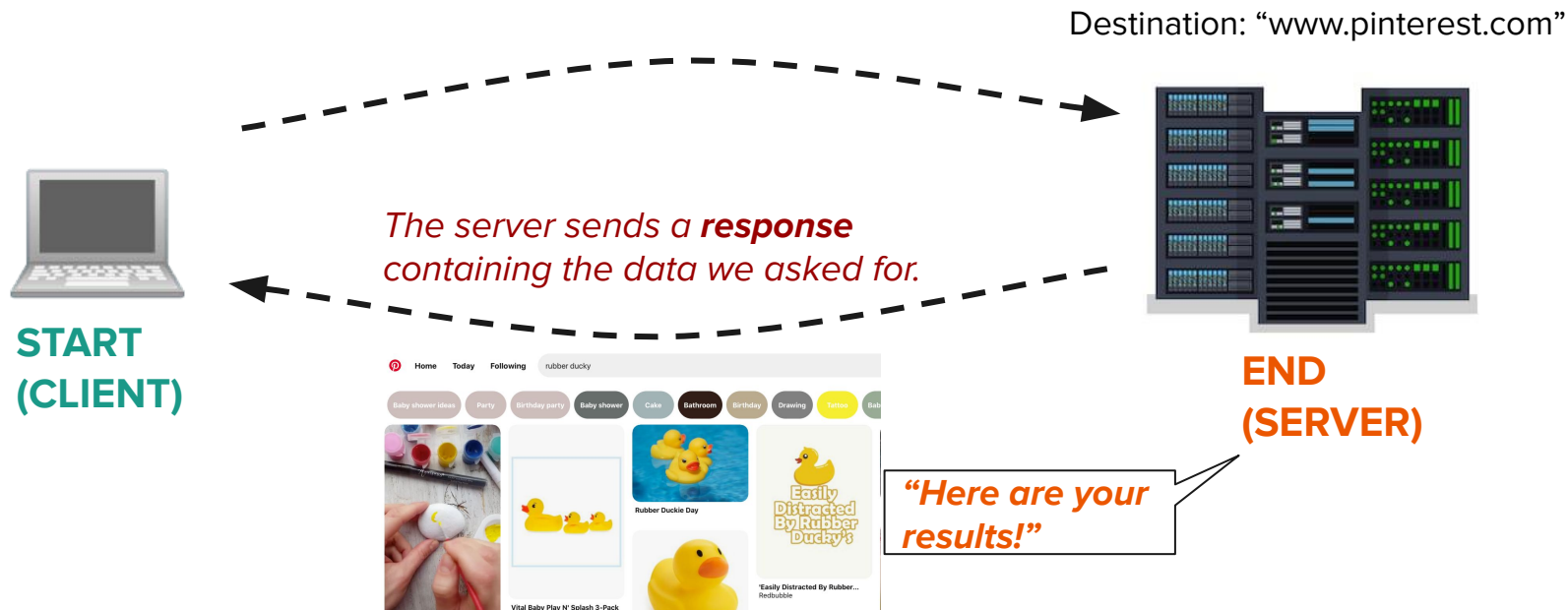
However, an **Internet route** (unlike a road trip) is always a **round trip!!!** Every time we make a **request** for data, the server sends a **response**.



We can also **send data** in our request - for example, a search query, or a username & password.



We can also **send data** in our request - for example, a search query, or a username & password.



Check for understanding

In the chat, rate your understanding by typing in a number from 1 to 5:

1

2

3

4

5

Activity (15 minutes)

In a group of 3:

- Open [this worksheet document](#) (don't make a copy).
- Choose a website to visit. (*E.g. Pinterest.com*)
- Write down **3 different actions** you can take on the site. (*E.g. Search for a keyword, save a pin, visit a pin*)
- For each action, write down:
 1. What is the request? What data are you sending?
 2. What is the response? What is being sent back to you?

My first web application!

What is Flask?

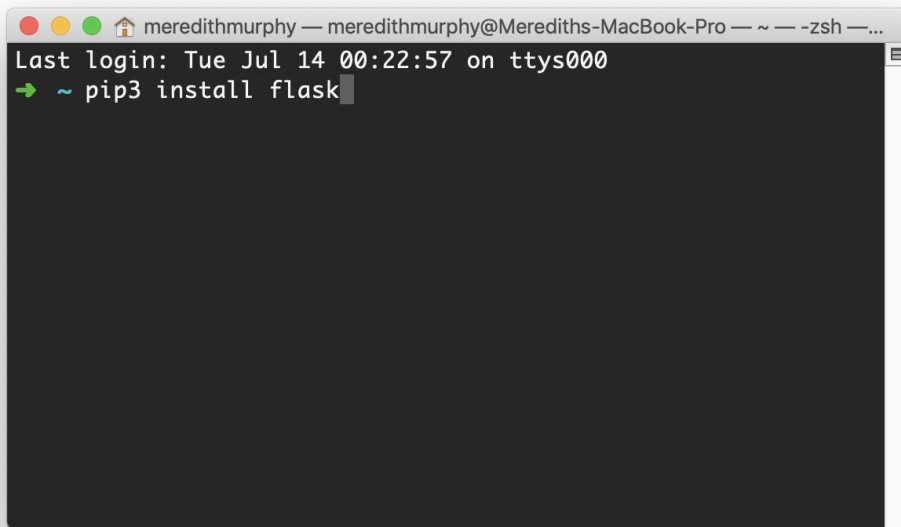
In this course, we'll be using the **Flask** web framework. Flask is called a **microframework** because it does not require other tools or libraries.

Flask will provide us with the tools we need to create our first web application!

Let's install Flask so that we can use it to build our first web app.



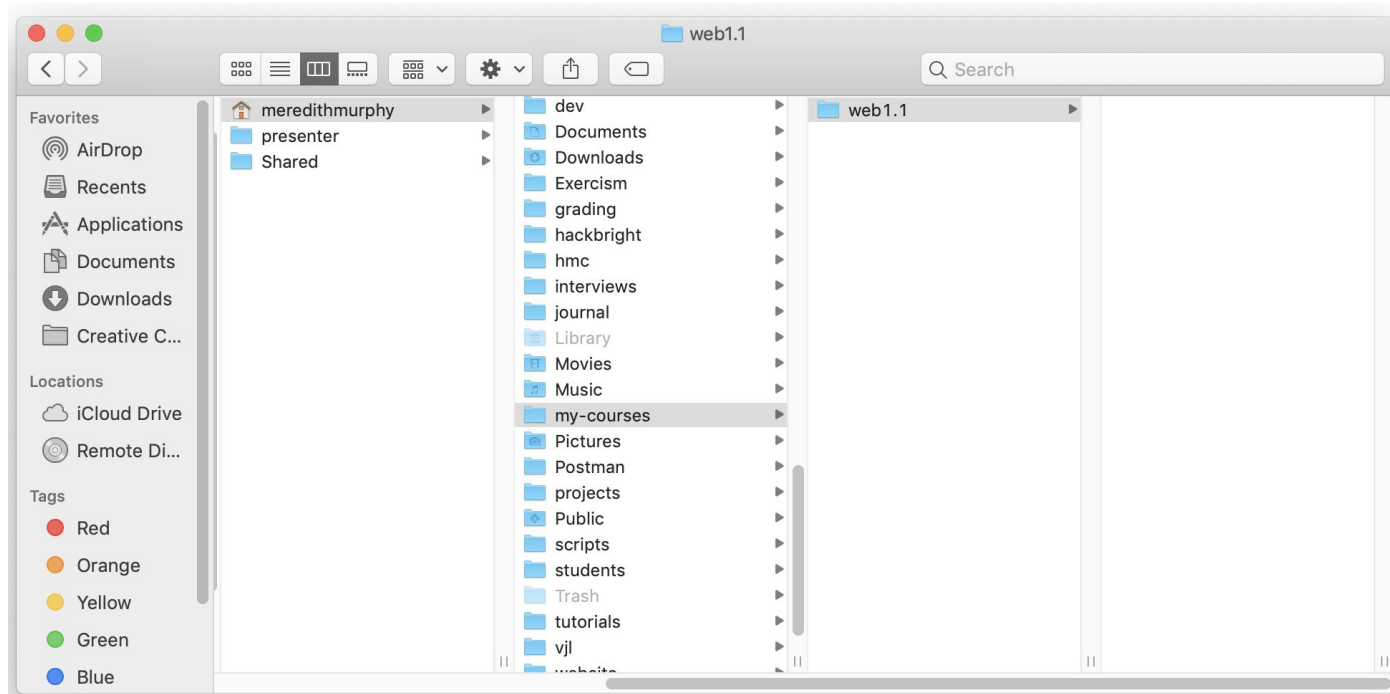
Open up a Terminal and type in “pip3 install flask”.



```
meredithmurphy — meredithmurphy@Merediths-MacBook-Pro — ~ — -zsh —...  
Last login: Tue Jul 14 00:22:57 on ttys000  
➔ ~ pip3 install flask
```

If this command doesn't work, make sure you've installed Python with “brew install python3”.

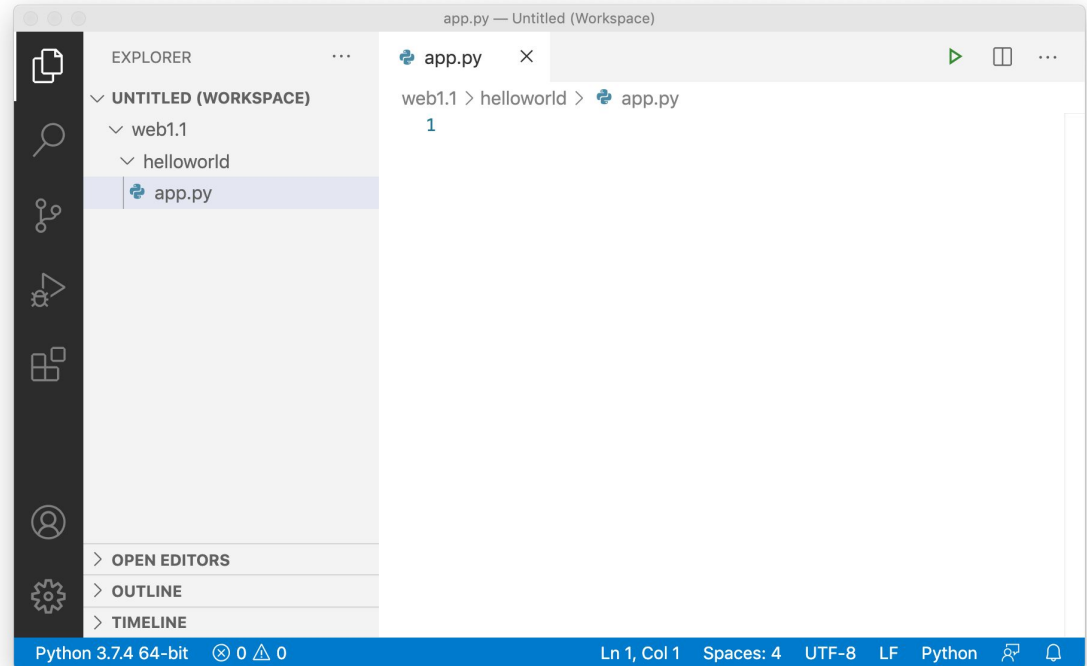
Go to **Finder** and create a folder for this class. (Should be in the “dev/courses” folder.)



Hello, World!

Go to **VSCode** and select File -> Add Folder to Workspace. Then do the following:

- Add your class folder
- Create a new folder in it called “helloworld”
- Create a new file in that called “app.py”



Type the following into “app.py”. Make sure you copy over everything exactly.
(Capitalization, punctuation, etc matters!)

```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def homepage():
    return "Hello, world!"

if __name__ == '__main__':
    app.run(debug=True)
```

Now, let's add a route function:

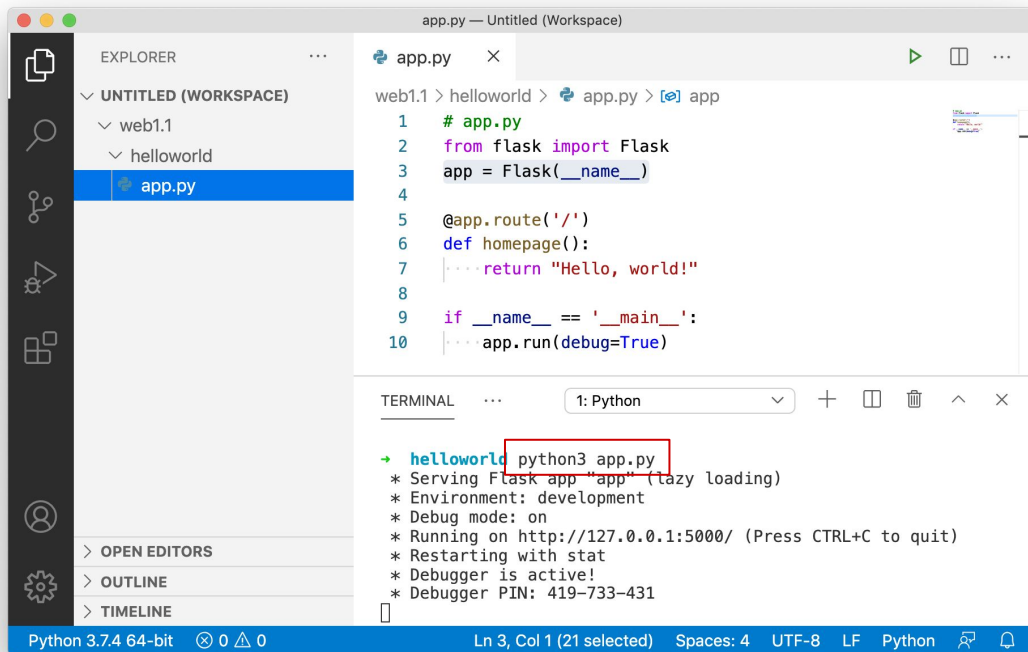
```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def homepage():
    return "Hello, world!"

if __name__ == '__main__':
    app.run(debug=True)
```


Hello, World!

Right-click on “helloworld” in the folder tree and click “Open in Terminal”. Then type in “python3 app.py” to run.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a workspace with a folder named 'web1.1' containing a subfolder 'helloworld' and a file 'app.py'. The 'app.py' file is selected. The main editor window displays the code for 'app.py', which is a simple Flask application. The code includes imports for Flask, a route for the homepage, and a main block to run the application with debug mode enabled. The terminal at the bottom shows the command 'python3 app.py' being executed, with the output indicating that the Flask app is running on http://127.0.0.1:5000/.

```
app.py — Untitled (Workspace)

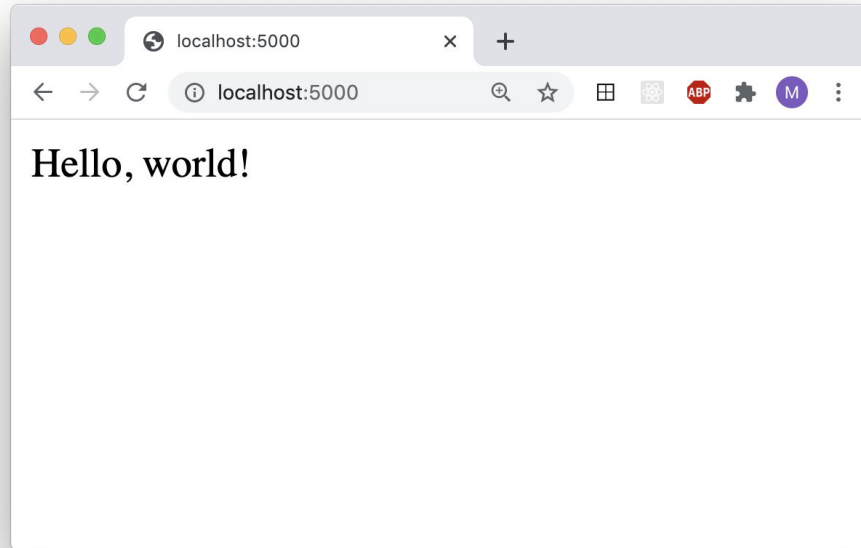
EXPLORER
  UNTITLED (WORKSPACE)
  web1.1
    helloworld
      app.py

app.py
1 # app.py
2 from flask import Flask
3 app = Flask(__name__)
4
5 @app.route('/')
6 def homepage():
7     return "Hello, world!"
8
9 if __name__ == '__main__':
10     app.run(debug=True)

TERMINAL
1: Python
→ helloworld python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: development
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 419-733-431
```

Hello, World!

Open any browser and type “localhost:5000” into the URL bar. You should see our message!



Break - 10 min

So... What just happened here?!?

What just happened?

Let's see if we can deconstruct the “app.py” code.

```
# app.py
from flask import Flask # Import the Flask library for use in our routes
app = Flask(__name__) # Create an “app” variable to use for route creation

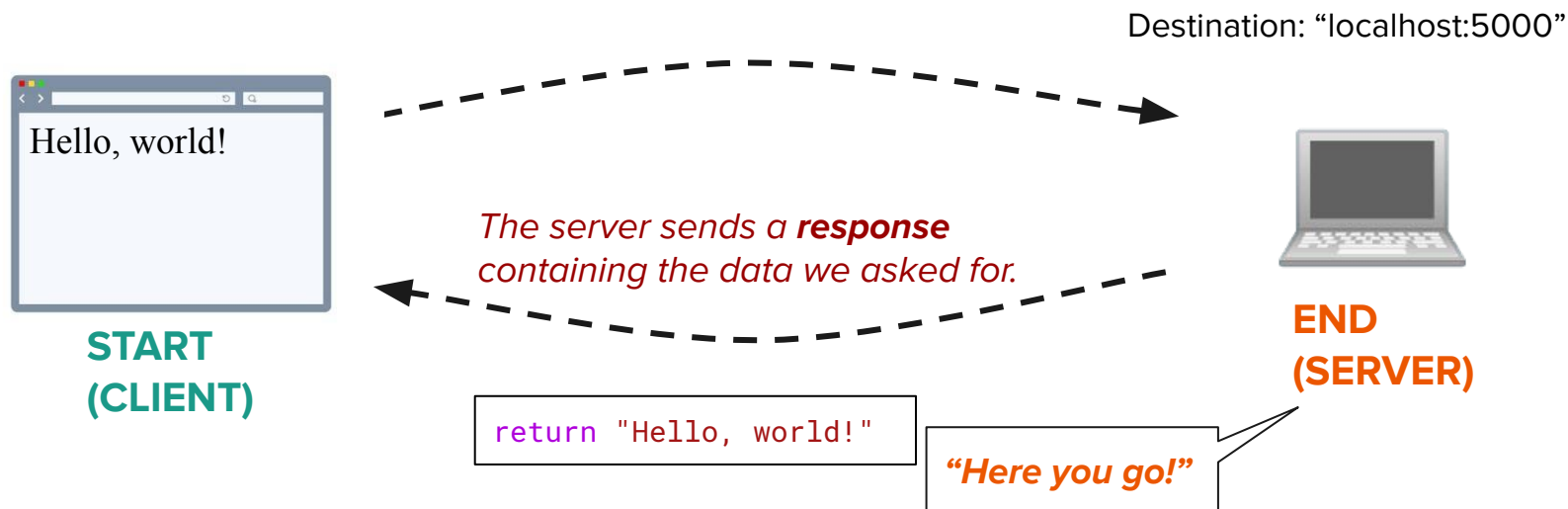
@app.route('/') # The following function will be run for the URL “/”
def homepage(): # Define a function “homepage”
    return "Hello, world!" # Return a response of “Hello, world!”

if __name__ == '__main__': # If this is the file being run
    app.run(debug=True) # Run the web application in debug mode
```

By running the Flask application, we now have a **server** running on our computer. So, we're both **making the request** (in the browser) *and* **sending the response** (from the server)!!!



By running the Flask application, we now have a server running on our computer. So, we're both **making the request** (in the browser) *and* **sending the response** (from the server)!!!



Check for understanding

In the chat, rate your understanding by typing in a number from 1 to 5:

1

2

3

4

5

Activity: Make Another Route (10 minutes)

In a group of 3, **write another route for a different URL**. Give the page some content. Get creative! It could be...

- A profile page for your dog
- A description of your favorite movie
- Etc...

You can also add HTML! Try adding an HTML tag in the returned value, and see if it appears on the page.

Route variables, or: How to make Infinite Web Pages

Last time, we created some **route functions** to serve various web pages. We ended up with something like this:

```
# app.py
from flask import Flask
app = Flask(__name__)

@app.route('/profile/Rey')
def my_page():
    return "<h1>Hello, Rey!</h1>"

if __name__ == '__main__':
    app.run(debug=True)
```

BUT... that gives us **only one web page!** What if I need a profile page for every single person here?! Man, that'd take a while...

```
...

@app.route('/profile/Luke')
def luke_profile():
    return "Hello, Luke!"

@app.route('/profile/Chewbacca')
def chewbacca_profile():
    return "Hello, Chewbacca!"

@app.route('/profile/Padme')
def padme_profile():
    return "Hello, Padme!"
```

```
@app.route('/profile/Leia')
def leia_profile():
    return "Hello, Leia!"

@app.route('/profile/Lando')
def lando_profile():
    return "Hello, Lando!"

@app.route('/profile/Jar-Jar')
def jarjar_profile():
    return "Hello, Jar-Jar!"

...
```

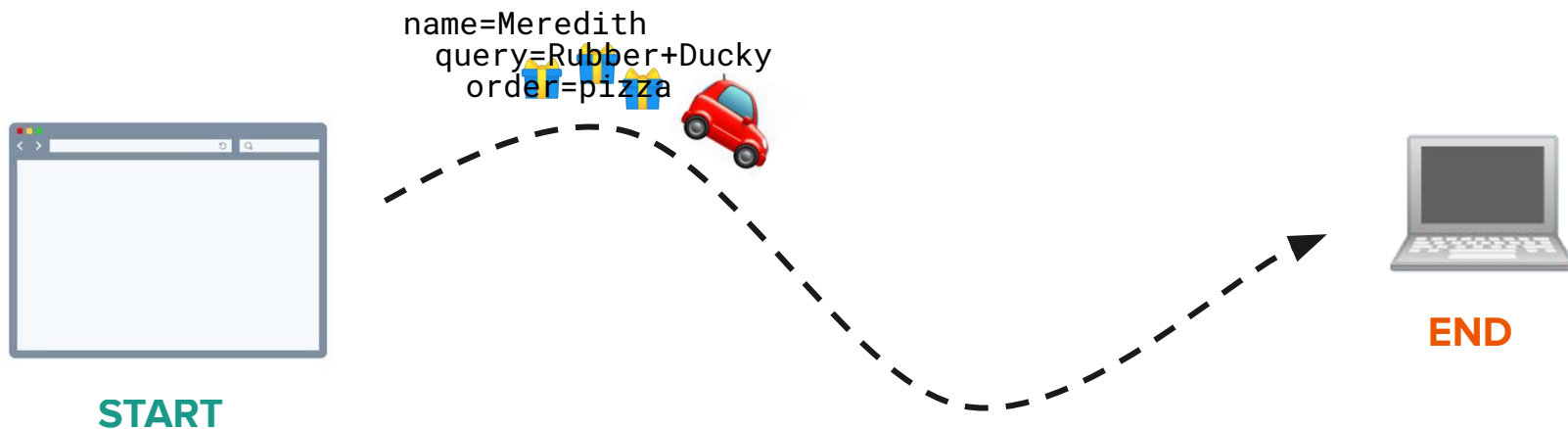
It turns out, there is a way to do that! We can make 5, 10, 20 web pages all at once... or even *infinite* web pages! Check out [this page](#) for an example (try typing your name into the URL)!

So... how do we do it?!?

Route Variables!!!

A **route variable** is an extra piece of data that is sent along the route by the **client** (your browser) to the **server** (in this case, your laptop).


Route variables are always sent as **key-value pairs**.



Open up the `app.py` file from last class and add the following route:

```
@app.route('/profile/<users_name>')  
def profile(users_name):  
    return "Hello " + users_name
```

Whoa - we just made infinite URLs! With only 3 lines of code!!!!

 What are **3 things you notice** about the `users_name` variable? Type into the chat!

What is this class all about, anyway?

Go over the [course syllabus](#) together & answer any questions.

Lab Time

Reading 1: Due Sunday night on Gradescope

Homework 1: Due next Thursday night

Stay in the main Zoom room if you'd like to stay for more Q&A, homework help, etc.

Go to your individual breakout room if you'd prefer to have quiet time!