FLASK WEB DEVELOPMENT

WHAT IS FLASK?

- Flask is a Python web framework that makes it easy to create a fully-featured web application.
- Learn the basics of this popular framework so that you can create your own web application with a Python back-end.
- Flask is a micro web framework written in Python.
  - It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

INSTALLING LIBRARIES AND CONFIGURING YOUR ENVIRONMENT

SETTING UP OUR ENVIRONMENT

- Install Python 3.10+ and PIP
- Install virtualenv
- Crete our project directory
- Create virtual environment
- Install required packages

STEP #1: INSTALLING PYTHON AND UPDATING PIP

INSTALLING PYTHON 3.10+

- www.python.org/downloads
  - Note for WINDOWS users:
    - avoid using the default installation. Customize your installation directory to something like your documents folder
  - After installing python check if it has been installed correctly:
    - Open a new command/terminal window and type `python –version`
- Pip installation:
  - Pip comes with python installation, but it is recommended to update it:

```
python.exe -m pip install --upgrade pip
```

STEP #2: SETTING UP THE PROJECT ENVIRONMENT

SETTING UP THE PROJECT ENVIRONMENT

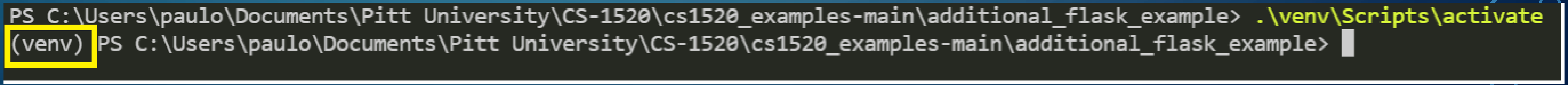
- Setting up a project directory:
  - From inside VS Code app, add a new folder (top left of the screen) “flaskintroduction”

SETTING UP THE PROJECT ENVIRONMENT

- Setting up a virtual environment (continuation):
  - Virtual environments allow you to have all your project configurations part of your project, not the VS Code environment
    - Great feature when collaborating your code with others ➔ when you send your project to others, the environment setup is sent to them too automatically
  - From within the project directory, open a new terminal window inside VS Code), and type: pip install virtualenv
  - From the terminal window type: virtualenv env
  - In the explorer tab in VS Code you should see now an "env" directory

SETTING UP THE PROJECT ENVIRONMENT

- Activating the environment:
  - MAC OS: type: source env/bin/activate
  - Windows:
    - It is a little bit more complicated:
      - Follow instructions found in the following page, but read the page contents carefully:
        - https://www.repairwin.com/fix-running-scripts-disabled-on-windows-10/
    - Then go back to the VS Code terminal and type: .\\venv\Scripts\activate
- Checking if the env has been activated:
  - You should see an (env) at the beginning of your prompt command



```
PS C:\Users\paulo\Documents\Pitt University\CS-1520\cs1520_examples-main\additional_flask_example> .\venv\Scripts\activate
(venv) PS C:\Users\paulo\Documents\Pitt University\CS-1520\cs1520_examples-main\additional_flask_example>
```

INSTALLING THE PROJECT REQUIRED PACKAGES

INSTALLING THE PROJECT REQUIRED PACKAGES

- Installing flask and flask-sqlalchemy
  - Within VS Code terminal window with the (env) showing in the command prompt type:

```
pip install flask flask-sqlalchemy
```

CREATING OUR FIRST FLASK WEB APPLICATION

CREATING OUR FIRST FLASK WEB APPLICATION

- Create the file main_app.py in the current folder
- Add the following lines into it

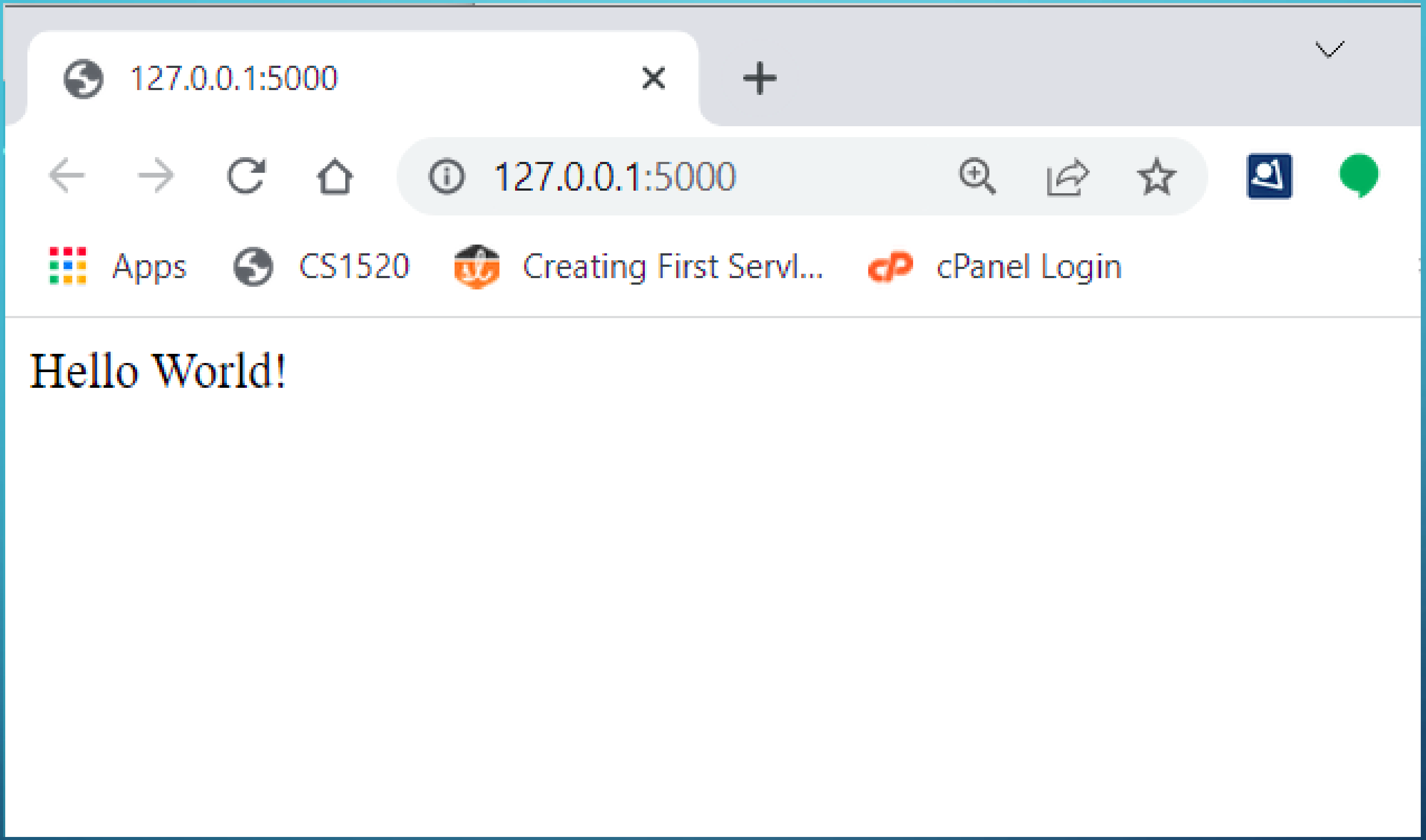
```
from flask import Flask
app = Flask(__name__)

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

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

HOW TO RUN THIS FLASK APPLICATION

- In the VS Code terminal window:
  - First check if you are with the environment running
    - (env) PS C:\Users\paulo\Documents\Pitt University\CS-1520\cs1520_examples-main\flask_codes_for_lecture>
  - Type `python main_app.py`

```
 * Serving Flask app 'main_app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a …
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C)
```

HOW TO RUN THIS FLASK APPLICATION



Hello World!

ADDING ENDPOINTS TO OUR MAIN APPLICATION

ADDING ENDPOINTS TO OUR MAIN APPLICATION

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

@app.route("/foo")
def fooController():
    return "<h1>THIS IS THE FOO PAGE</h1>"

@app.route("/bar/")
def bar():
    return "<h1>this is the bar page</h1>"
```

GENERATE HTML PAGES FROM ENDPOINTS

THE PAGE FOR THE GET REQUEST

```
formpage = """<!DOCTYPE html>
<html>
    <head>
        <title>Basic form</title>
    </head>
    <body>
        <form action="" method="post">
            Enter a number:  <input type="text" name="anumber" />
            <br />
            Enter a string:  <input type="text" name="astring" />
            <br />
            <input type="submit" value="submit" />
        </form>
    </body>
</html>
"""
```

THE PAGE FOR THE POST REQUEST

```
presentpage = """<!DOCTYPE html>
<html>
    <head>
        <title>Present data!</title>
    </head>
    <body>
        You entered this number:  {}
        <br />
        You entered this string:  {}
    </body>
</html>
"""
```

THE HOME PAGE

```
@app.route("/", methods=['GET', 'POST'])
def form():
    if request.method == 'POST':
        return presentpage.format(request.form["anumber"],
                                  request.form["astring"])
    else:
        return formpage
```

STUDYING THE LOGGING WEB APP

STUDYING THE LOGGING WEB APP

- Open up the login_version_1.py
- Try to go over all the code
- Demonstrate how each path would be reached from user input in the browser