Flask installation

1. Ensure python3 and pip are installed and up to date
2. Create a directory, in that directory create a virtual environment: python -m venv <environment name>
   1. May need to type python3 depending on OS
   2. Environment name can simply just be venv
3. Activate the environment: venv\Scripts\activate
   1. If using windows PowerShell will ned to run as administrator
   2. And possibly first run Set-ExecutionPolicy Unrestricted
4. Install flask: pip install flask
5. Can now import or build new flask app in this directory
6. Must run “FLASK\_APP=<name of main python file>” and then “flask run” after each change to code, open browser to whatever local address and port is reported in the command window to see changes

Directory Structure

1. At minimum must have a top-level python file (i.e. on the same level as the application directory) that defines which application to run
2. Optionally create a script file to automate the process of launching the flask app each time, also in the top-level directory
3. Inside the application directory (which can be called anything) need to at minimum have
   1. the \_\_init\_\_.py file, which will initialise the app variables
   2. the routes.py file which will define the weblinks and the methods associated with those links
4. A templates folder inside the application folder, which will hold the html files
5. Config.py in top level to hold configuration variables such as secret key and database connection details
6. requirements.txt, which will hold all the import names

Steps to creating new page

1. Create the html file in the templates folder
2. Use extends and block to reuse HTML code from other files
3. In routes.py define a new method with same name as the HTML file, use @main.route(‘/name’) to reference the file
4. Use render\_template(‘path\_to\_file’, optional\_passed\_parameters= optional\_passed\_parameters, …)
5. Use redirect to go to a different page than the current one
6. If passing in data by GET or POST, make sure to set that in method definition
7. If using flash, make sure it’s displayed in the HTML (see base.html)
8. Use url\_for instead of name of URL path

Passing data from python to html

1. Can pass strings, dictionaries
2. Flask uses Jinga2 to handle the variables on the HTML side

Passing data asynchronously (ajax)

1. Link up HTML data and javascript functionality as usual
2. Use ajax to send the serialised data to a method in the python file
3. On python side, use request function (make sure its imported) to get the data from the POST
4. Send back data from the python side with return
   1. Can use json.loads or json.dumps to return a string from a json file
5. Chain success and error methods to the ajax method in the javascript file to handle to returning data

HTML Manipulation

1. Use {{ … }} where … is the variable name
2. Can use {{% if <cond> %}}, {{% else %}}, {{% endif %}} for control conditions
3. Can use {{% for <variable name> in <passed in dictionary> %}}, {{% endfor %}} to iterate through a dictionary
4. Use the extends and block commands to reuse HTML text (see base.html and index.html)

Forms

1. Install wtf-forms to handle form processing: pip install flask-wtf (make sure you’re in the virtual environment!)
2. Forms are defined in python classes, create a python file forms.py in the application folder to hold different form types (login, registration, etc)
3. Import the different kind of fields from wtforms, strings, integers, Booleans, etc
4. Fields can be validated with wtforms.validators
5. Errors can be shown in the HTML with the form.(fieldname).errors property (see login.html)

Configurations

1. Create config.py in the top level to hold configuration details
2. Important one is “Secret Key” which is used to generate cryptographic keys
3. No need to render the fields, will be done automatically
4. Use the label attribute to display name of field
5. Use the () after the field name to display the actual field, can pass attributes as arguments here, like form.password(size=32)

Database

1. Ensure flask-mysql and PyMySQL are installed
2. Import pymysql in config.py
3. Create the create\_connection method
4. Put database details in the pymysql.connect function
5. In python files that will interact with database:
   1. Ensure MySQL and DictCursor are imported
   2. Ensure mysql is initialised
   3. Use cursor = create\_connection().cursor()
   4. Use the cursor to execute sql commands
   5. Pass the fetched data as a variable in the render\_template function