**MIT xPRO Data Engineering Certificate**

**Flask**

Flask is a Python library that allows developers to build web applications.

pip install flask

**What Is the Flask Web Server?**

Flask is a web framework. This means Flask provides you with tools, libraries, and technologies that allow you to build a web application. So far, you have started building simple applications on a local server on your machine (e.g., localhost/5000). However, you need to keep in mind that Flask can also be used for building more complicated applications for release on the Web.

As you have seen, the entirety of the code written in Flask is in Python. This means that basic Flask applications need very few dependencies of their own. The downside of this is that if you want to create more complex applications, you will need to independently research some of the available plug-ins to achieve more complex results.

**What You Need to Start Building a Flask Application**

As you have learned, Flask mainly relies on Python. For this reason, when you start using Flask to build a very simple website, such as the one you saw in the previous videos, all you need is a working installation of Python, and of course, a text editor, such as VS Code.

Before you begin to create a Flask application, a wise preliminary step is to create a folder on your machine where you will store all the needed files for that application. You can do this by opening your Terminal window and creating a folder using the command mkdir <folder\_name>.

Next, you will need to create a file, app.py, that will contain the code to build your application. Generally, this file has the following basic structure:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return 'Hello world'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0', port = 5000)

Note that the structure of the app.py file is used for creating a simple website that displays the words “Hello World”.

As you have learned in the previous videos in this module, you can also add requests, such as GET and POST, to make your HTTP requests more specific.

In the example file structure, “Hello World” is written as a simple string. To make your application more visually appealing, you can use HTML to write your text and add more details to your application, such as navigation bars, menus, and/or colors.

In order to include HTML code in your application, you will need to create a new folder titled templates, and insert a file called index.html in it. In this file, you will be able to include everything you want to be displayed as HTML in your application.

In order to render the HTML, you will also need to include the code render\_template in the first line of the file app.py.

As a last step, you will need to define a Python decorator to read the index.html file. The simplest syntax to achieve this is as follows:

@app.route('/')

def index():

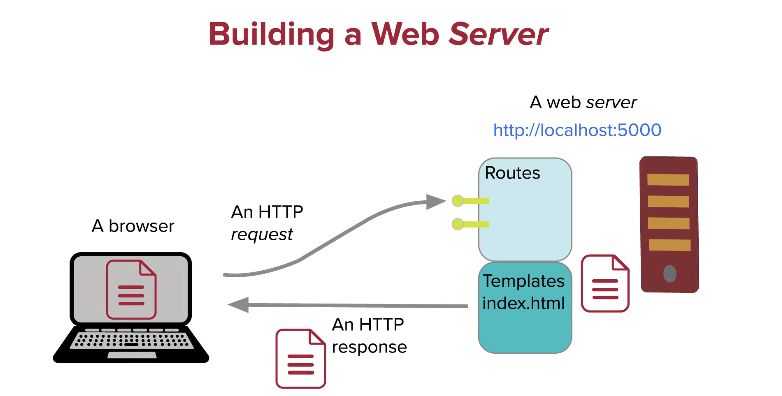
return render\_template('index.html')

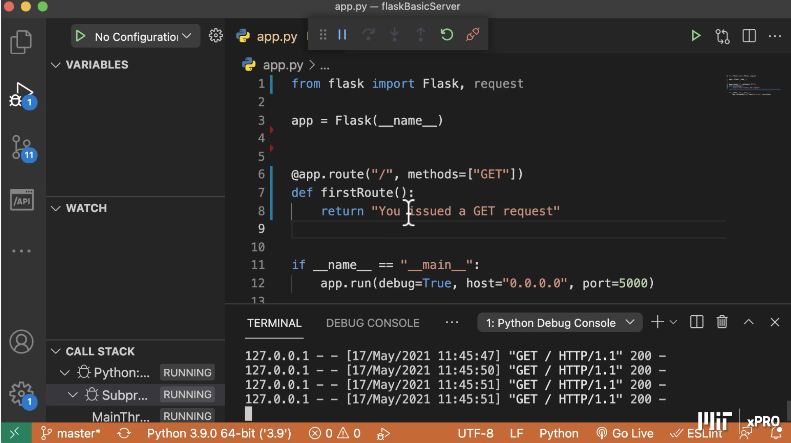
The code above makes Flask look for index.html in the templates directory in which the app.py program is in.

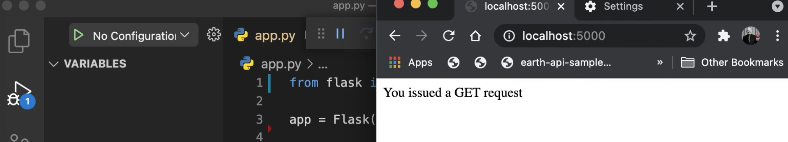
As you have learned in the previous videos, you can add as many files as you need in the template folder, based on what you want your application to include. The important thing to remember while doing this is to add the necessary decorators to the app.py file in order to render those templates.

**Jinja**

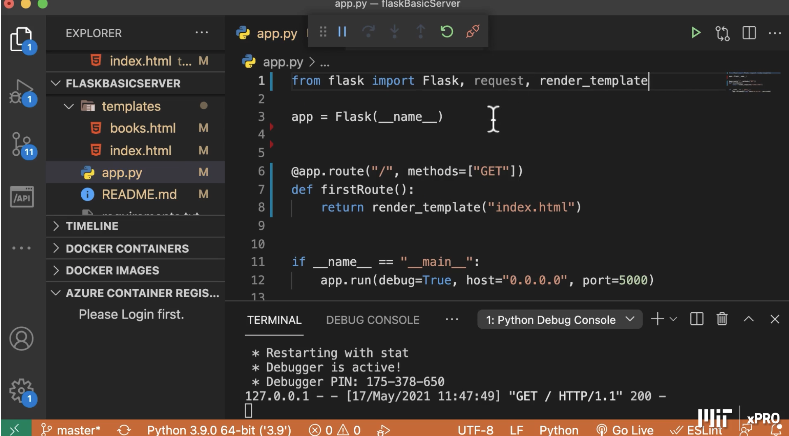
You can integrate Python and HTML together to produce web content using Jinja. Jinja is a templating language for Python developers that is used to create HTML format. It enables you to structure the pages that are passed to the browser.

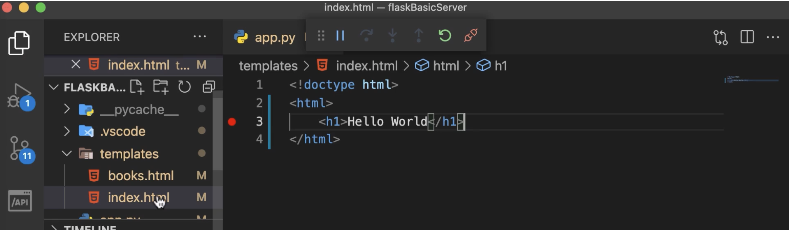


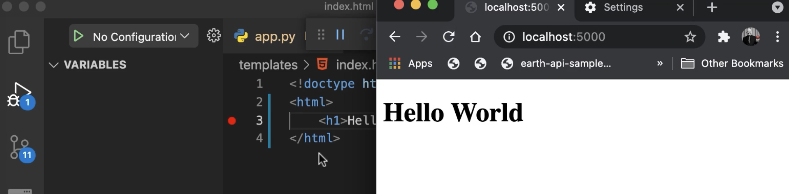


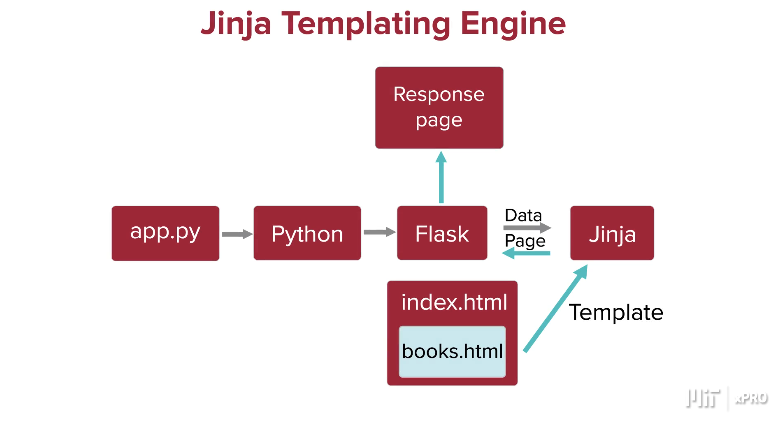


render\_template() is a flask function that looks up an html file and returns its contents to a web page:

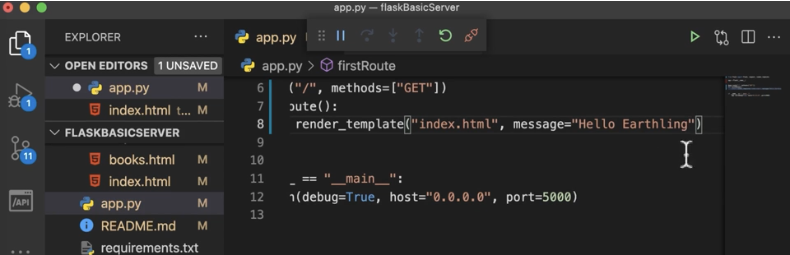


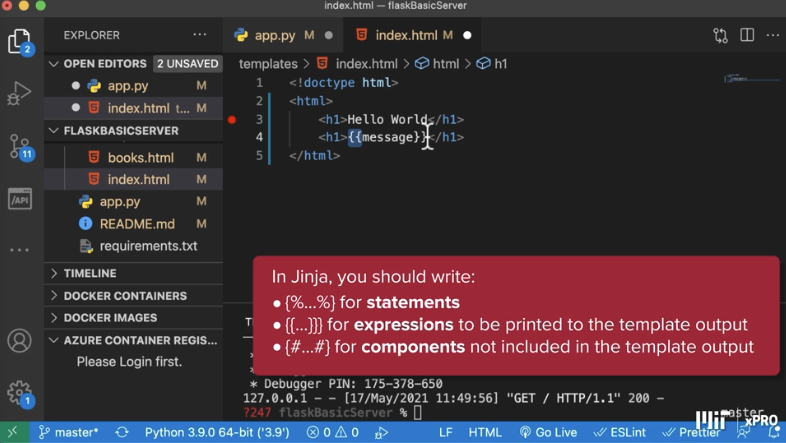


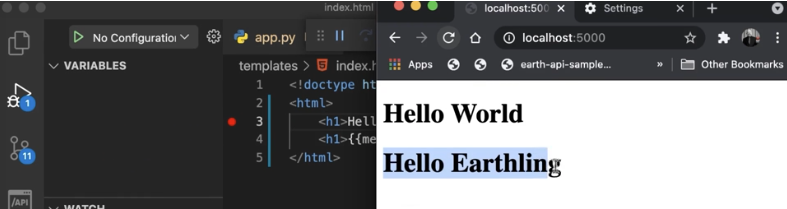


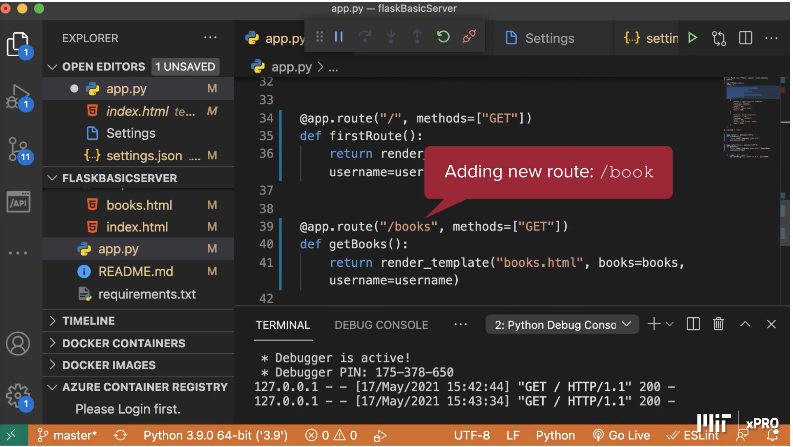
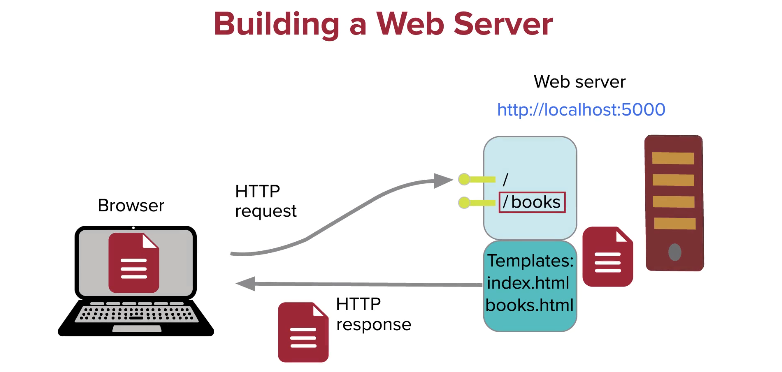


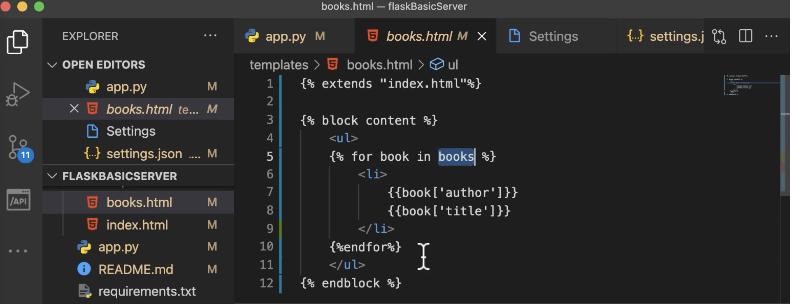
You can ask Flask to pass in data to be included in the html template output using Jinja, e.g. a message:

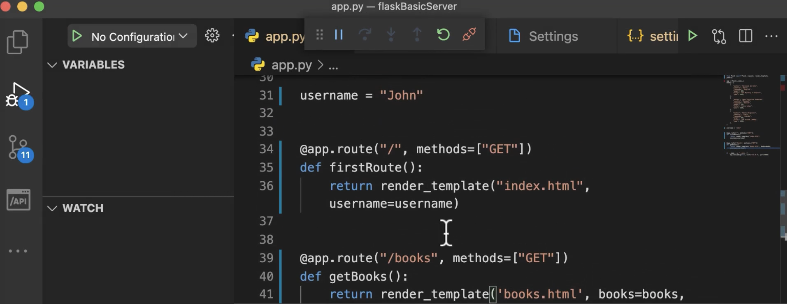


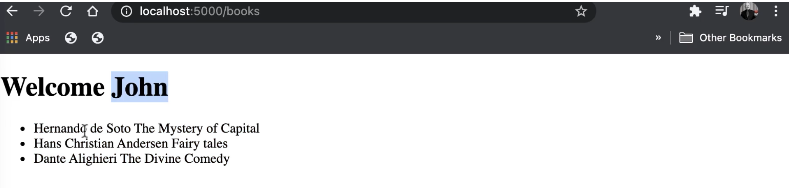


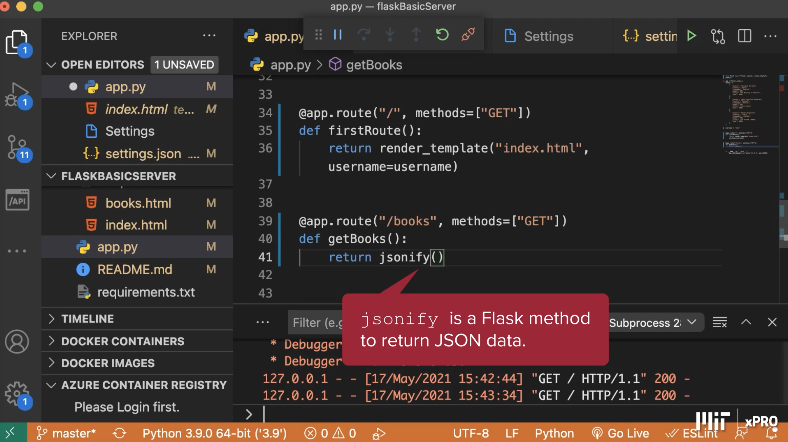


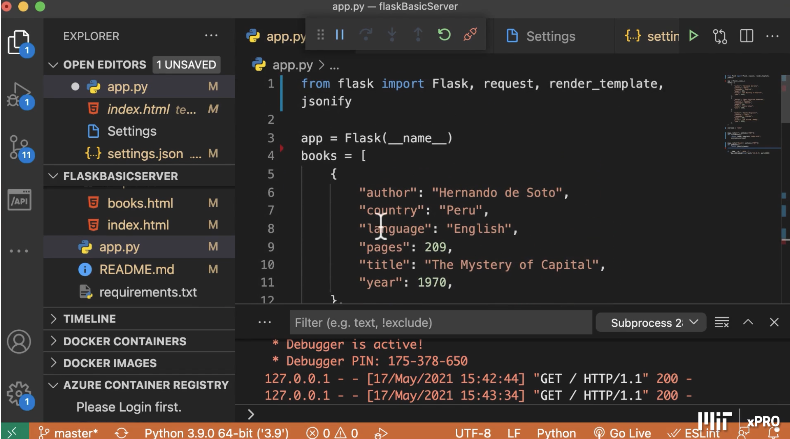




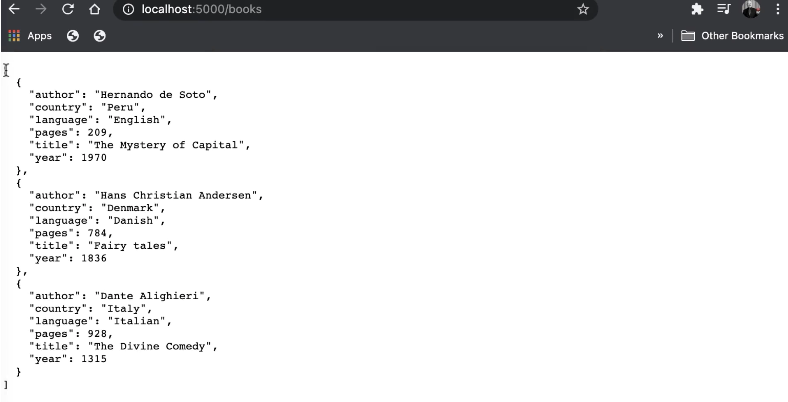




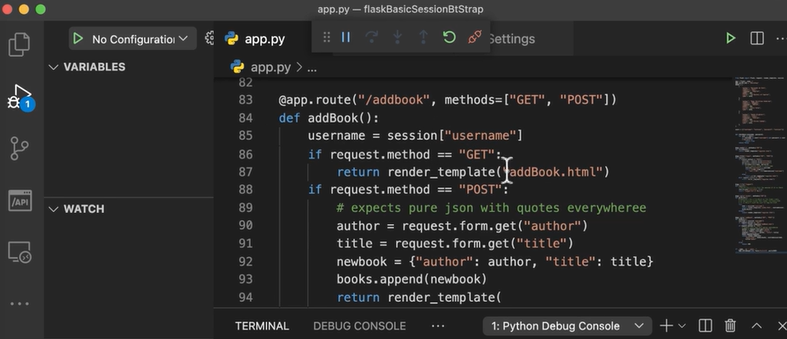




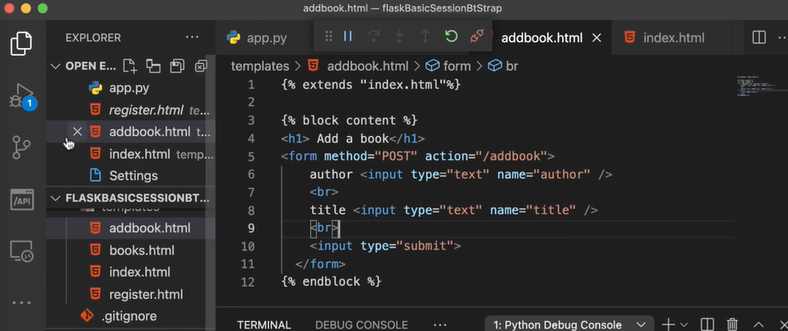
The books data is stored in a python dictionary. The jsonify() method in Flask will render the data from a Python dictionary into a JSON (JavaScript Object Notation) response. jsonify() takes a Python dictionary (and other JSON-serializable Python objects like lists, tuples, strings, integers, and booleans) and converts them into a Flask Response object.



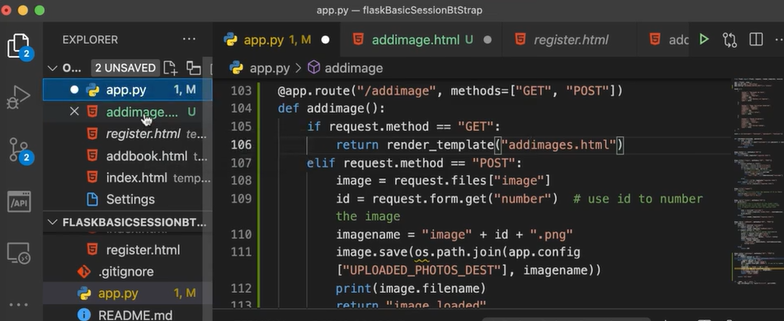
The code below adds a new route to add a book:



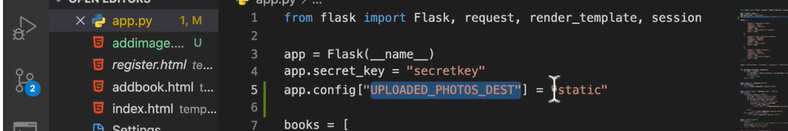
This uses the html template addbook.html:



This route is to add an image:

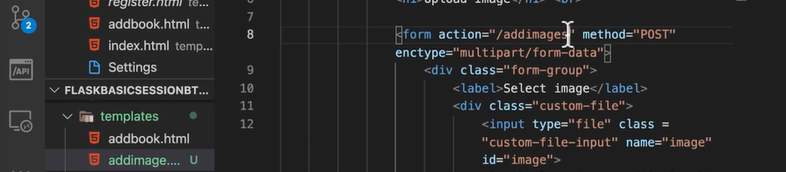


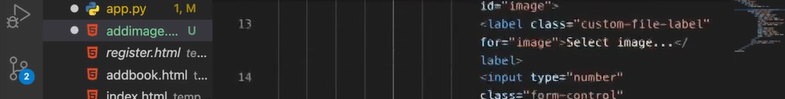
With a destination for where the images are uploaded:

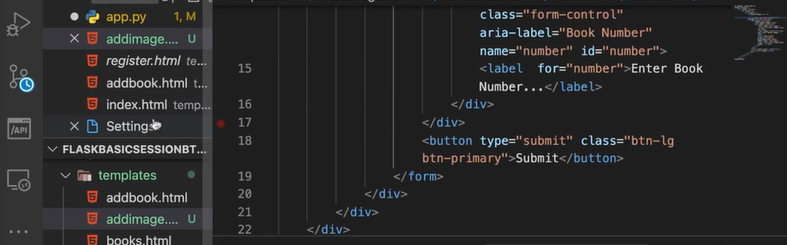


With this html template, which will allow you select an image and upload it:

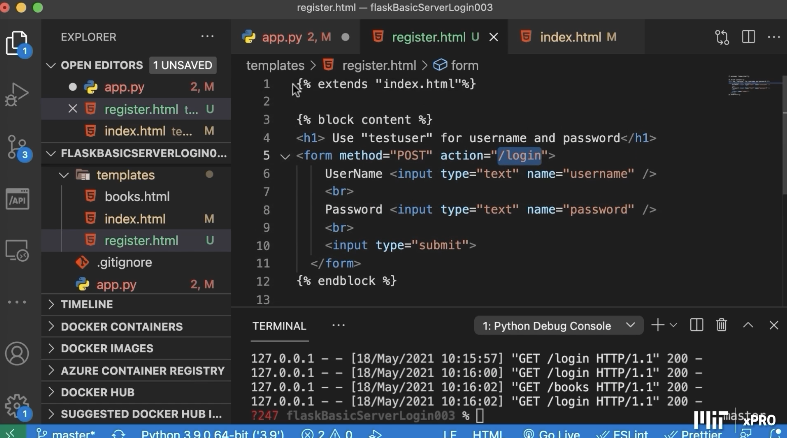


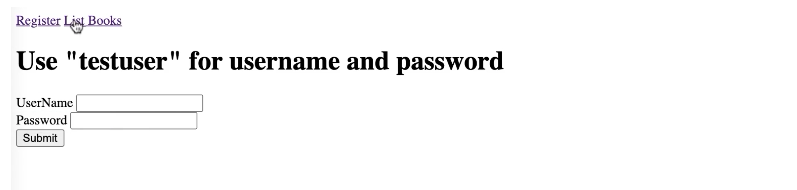




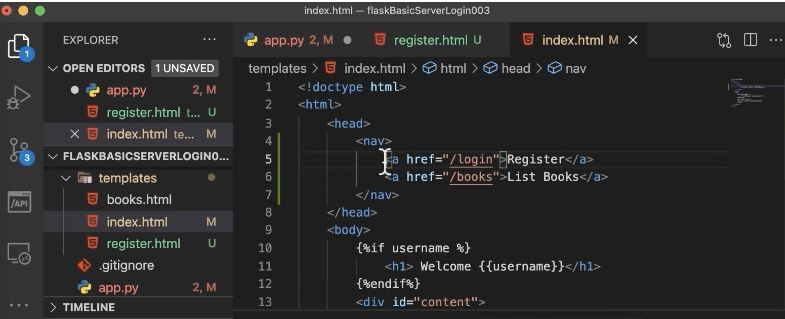


The register.html inherits and extends the index.html template, where the block content is replaced with that specified here:





This code adds in a navigation bar:



To display the books with the images, amend the books.html file to change how the list of books is rendered on the site:

