DBI Lab 009 - Python and Flask

COMP1048 - Databases and Interfaces

Dr Matthew Pike

> **Note** : You are not expected to complete all lab tasks in one session. You have three weeks to complete this lab sheet.

Setup

> **Note**: You should have done this already in Lab 001!

First, we must ensure that Python 3 and Flask are installed on your system. A brief outline of the process is outlined below, but is not exhaustive. Consult the following resources if the steps below do not work on your system:

- Python 3 Installation & Setup Guide
- Python 3 Download
- Flask Installation
- How can I install pip on Windows?

1. Ensure you have Python 3 installed, usually accessible using the command

   ```
   python3
   ```

   Alternatively, if this command does not resolve, try the following:

   ```
   python -V
   ```

   You should receive output similar too:

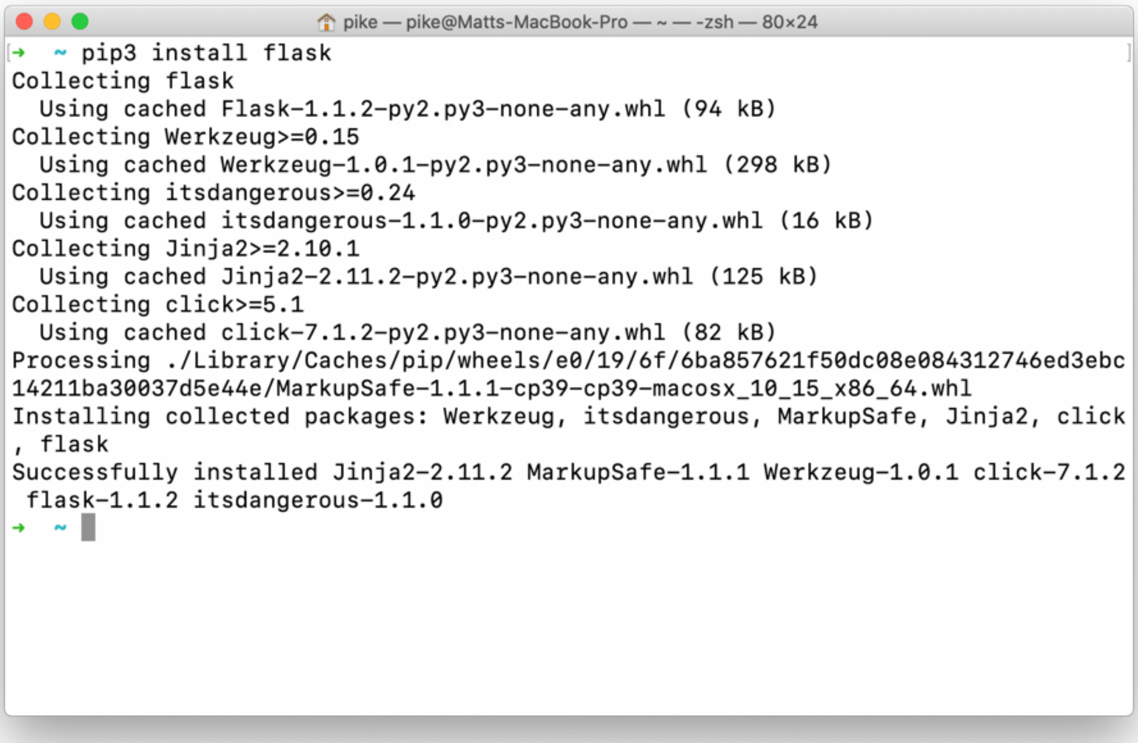
   ```
   Python 3.9.0
   ```

   (Your version number may vary. As long as it begins with `3.` then you have Python 3 installed.

2. Ensure that Flask is installed:

   ```
   pip3 install flask
   or
   pip install flask
   ```

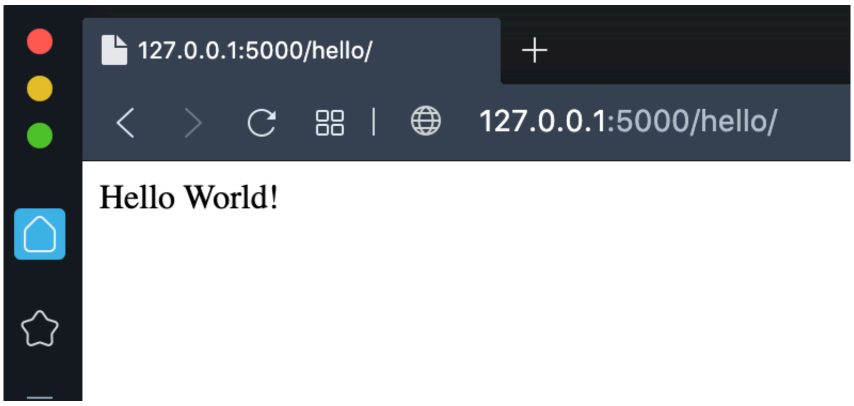
   Ensure that the command completes without error. Your console should look similar that shown below:



```
➜  ~ pip3 install flask
Collecting flask
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting Werkzeug>=0.15
  Using cached Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Collecting itsdangerous>=0.24
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting Jinja2>=2.10.1
  Using cached Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting click>=5.1
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Processing ./Library/Caches/pip/wheels/e0/19/6f/6ba857621f50dc08e084312746ed3ebc
14211ba30037d5e44e/MarkupSafe-1.1.1-cp39-cp39-macosx_10_15_x86_64.whl
Installing collected packages: Werkzeug, itsdangerous, MarkupSafe, Jinja2, click
, flask
Successfully installed Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2
 flask-1.1.2 itsdangerous-1.1.0
➜  ~ 
```

Task 1 - Say "Hello World!"

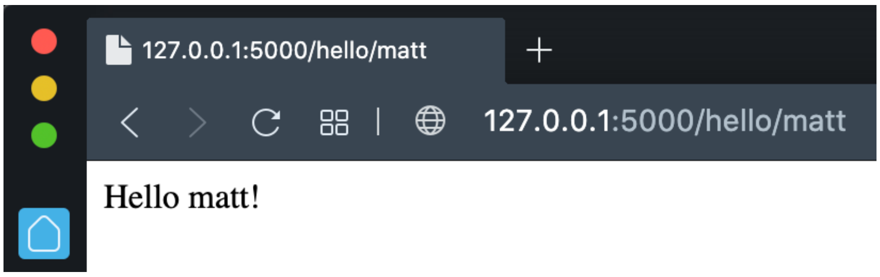
Following the steps set out in the lecture notes and the examples provided on Moodle, create a function that simply returns "Hello World!" when the user navigates to http://127.0.0.1:5000/hello/ . There is no need for a template or CSS styling, simply return the text "Hello World!". Your result should look like that shown below:



Hello World!

Task 2 - Make it Personal

Extend your solution to Task 1 to greet the user by name. The name should be supplied as a part of the URL. For example: http://127.0.0.1:5000/hello/matt , should output the following:



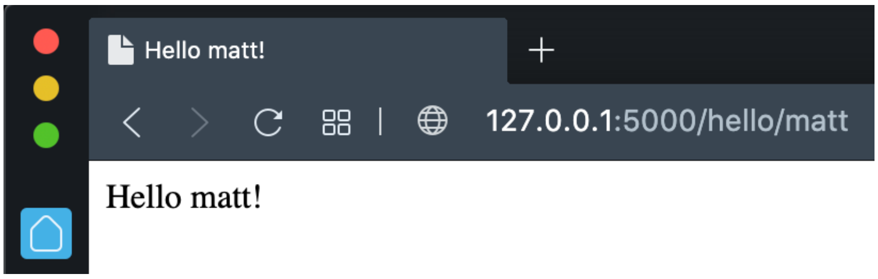
```
Hello matt!
```

Task 3 - Template it

Modify your solution to Task 2, such that the rendering of the provided `name` is done in a template rather than directly being returned as text from the function.

> **Remember**
>
> To use the templating functionality, you must import `render_template` from the Flask module, into the current namespace. This can be achieved using:
>
> ```
> from flask import (Flask, render_template)
> ```
>
> Also, remember that your template must reside in a subdirectory named `templates`. Your directory should have a structure similar too:
>
> ```
> ├── Lab9.py
> └── templates
>   └── hello.html
> ```

Your solution should look similar as the output for Task 2, with the only difference being the title of the HTML page also greets the user.



```
Hello matt!
```

Task 4 - Make it Pretty!

Create a folder to store a CSS file (remember this folder should be named `static`). Add some basic rules to a your CSS file and link it inside the template you made in Task 3.

> **Remember**

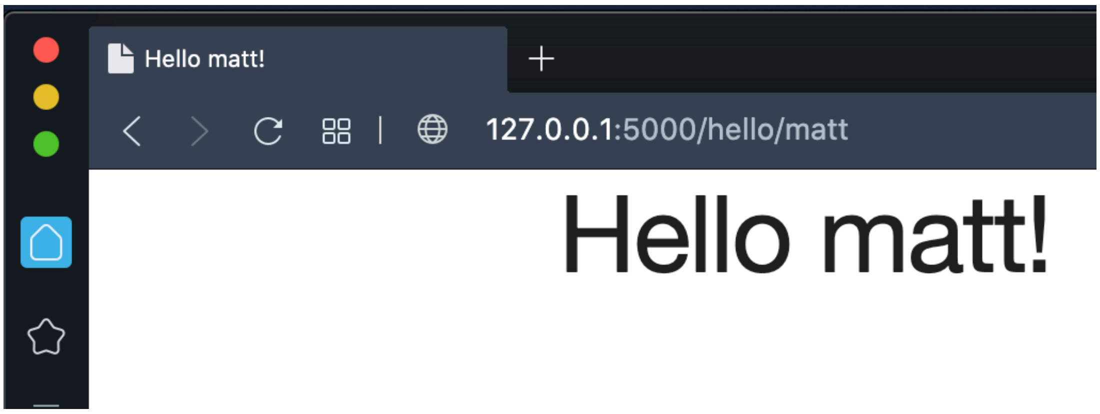
To use static assets you must import `url_for` from the Flask module, into the current namespace. This can be achieved using:

```
from flask import (Flask, render_template, url_for)
```

Also, remember that your static assets must reside in a subdirectory named templates. Your directory should have a structure similar too:

```
├── Lab9.py
├── static
│   └── style.css
└── templates
  └── hello.html
```

The CSS rules themselves are not so important, the key deliverable here is that the static resource is successful linked to inside our template file.



Hello matt!

Task 5 - Retrieving Items from a Database

Write the SQL necessary to represent the following `People` table in an SQLite database (named `People.db`):

ID	Name	Age
1	Alex	25
2	Bob	30
3	Charlie	35

Once complete, specify a `/people/` route in Flask, the logic for which should return all people listed in your `People` table. You should utilise a new template to achieve this. You should also add a "Add a New Person" link below the table which should redirect to `/people/add/` (we'll implement this in Task 6).

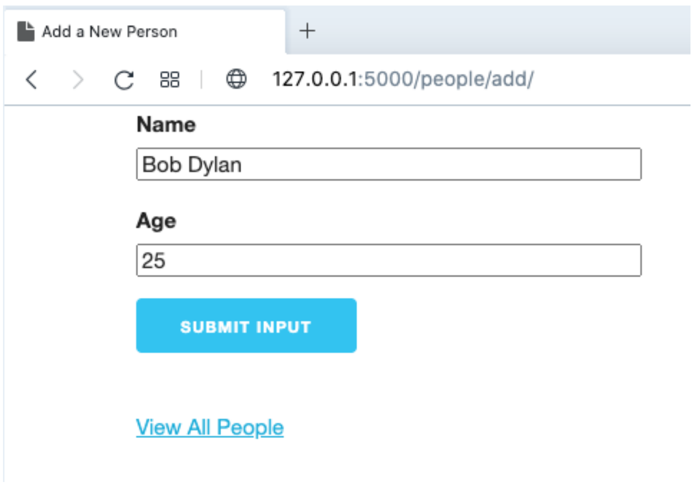
Remember

To interact with SQLite databases in Python, we must include the `sqlite3` module in our program's namespace. The following `import` statement achieves this:

```
import sqlite3
```

Task 6 - Add Items to a Database

Finally, add a route to `/people/add/` in Flask and specify the necessary interface and logic that allows users to insert new people into the `People.db`. You should redirect the user to `/people/` after the new person is successfully inserted into the database. If there is an error, redirect the user back to `/people/add/`.



Name

Bob Dylan

Age

25

Submission

Please submit a ZIP file containing solutions to each exercise. Solutions should be in subdirectories labelled according to their task number and contain all the neccesary files to run the solution.

Submitting this assignment will contribute 3% to your overall Module grade. Your submission should demonstrate reasonable effort and fulfil the specified requirements set out in this lab sheet in order to receive the full marks.

There is no granularity to the marking, the marking is on a pass-or-fail basis.

Registration is reported to Faculty office on a weekly basis. The submission point is available on Moodle.

Submission Deadline - Friday, 17 December 2021 @ 17:00