

# Lab Week 5

Integrating SQL into the Backend

Chaufournier & Wood  
CSCI 2541

Previously we've discussed...

# Creating a backend with python

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

@app.route('/')
def hello_world():
    return 'Hello, World!'

app.run(host='0.0.0.0', port=8080)
```

# SQL queries

- Create table queries

```
CREATE TABLE visitor_logs (ip_address text, visit_date text, address text);
```

- Insert queries

```
INSERT INTO visitor_logs VALUES ("127.0.0.1", "04/17/2020", "google.com");
```

- Select queries

```
SELECT * FROM visitor_logs WHERE ip_address = "127.0.0.1";
```

**Now we're going to combine  
them!**

# Integrating sqlite into python

Import the sqlite library

Create a new connection to the database

Create a cursor. It's a pointer to the database and tracks the location of operations.

The query we want to execute on the database.

Commit our changes to the database, so they are persisted.

Close the connection to save memory

```
import sqlite3

conn = sqlite3.connect('example.db')

c = conn.cursor()

c.execute('''CREATE TABLE stocks
          (date text, trans text,
           symbol text, qty real,
           price real)''')

conn.commit()

conn.close()
```

# Inserting data to a table

Insert queries use a special syntax

We use a concept called parameter substitution to specify the values

We separate the query from the input data itself and instead provide it as a tuple argument

```
import sqlite3

conn = sqlite3.connect('example.db')

c = conn.cursor()

c.execute("INSERT INTO stocks VALUES
(?, ?, ?, ?, ?)",
('2006-01-05', 'BUY', 'RHAT', 100, 35.14))

conn.commit()

conn.close()
```

# Inserting data to a table

Insert queries use a special syntax

We use a concept called parameter substitution to specify the values

We separate the query from the input data itself and instead provide it as a tuple argument

We do this as a security precaution. NEVER allow untrusted input into your database.

```
import sqlite3

conn = sqlite3.connect('example.db')

c = conn.cursor()

c.execute("INSERT INTO stocks VALUES
(?, ?, ?, ?, ?)",
('2006-01-05', 'BUY', 'RHAT', 100, 35.14))

conn.commit()

conn.close()
```



# Selecting data from a table

Select queries work just like any other query.

We use parameter substitution again to prevent XSS

We then have to fetch results from the cursor. To fetch one at a time we use the `fetchone()`

We check if `None` is returned meaning no matches.

If there is a match you receive a tuple. You can print out using tuple syntax.

```
c.execute('SELECT * FROM stocks WHERE symbol=?', ('RHAT'))
```

```
results = c.fetchone()
```

```
if results is None:  
    return
```

```
# ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)  
print(results[0], results[1])
```

# Selecting data from a table

If you expect a lot of results you can iterate over the cursor.

You receive back a row tuple that you can operate on.

If the result set is empty nothing is iterated over.

```
for row in c.execute('SELECT * FROM stocks
ORDER BY price'):
    print(row)
# ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)
```

# Integrating with Flask.

Flask is just like any other python code.

Add the database logic in the specific endpoint

You need to create a new connection in every endpoint due to threading issues in python

Make sure you close your connection at the end to save memory!

```
from flask import Flask
import sqlite3

app = Flask('app')

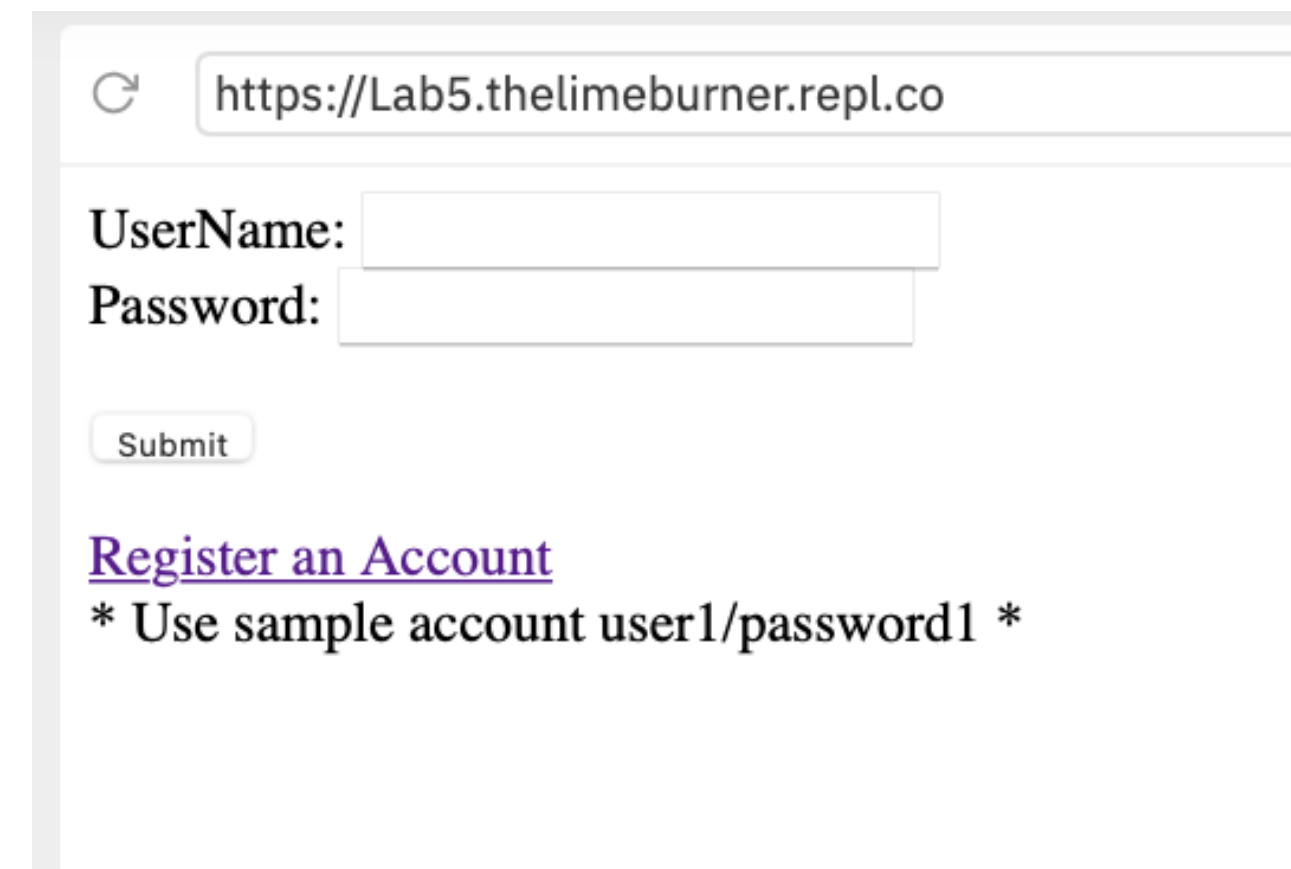
@app.route('/')
def index():
    conn = sqlite3.connect('example.db')
    c = conn.cursor()
    c.execute('SELECT * FROM stocks WHERE
              symbol=?', ('RHAT',))
    results = c.fetchone()
    if results is None:
        return 'Not Found'
    conn.close()
    # ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)
    return results[1]+" "+results[2]+" on "+
           +results[0]+" for "+results[4]

app.run(host='0.0.0.0', port=8080)
```

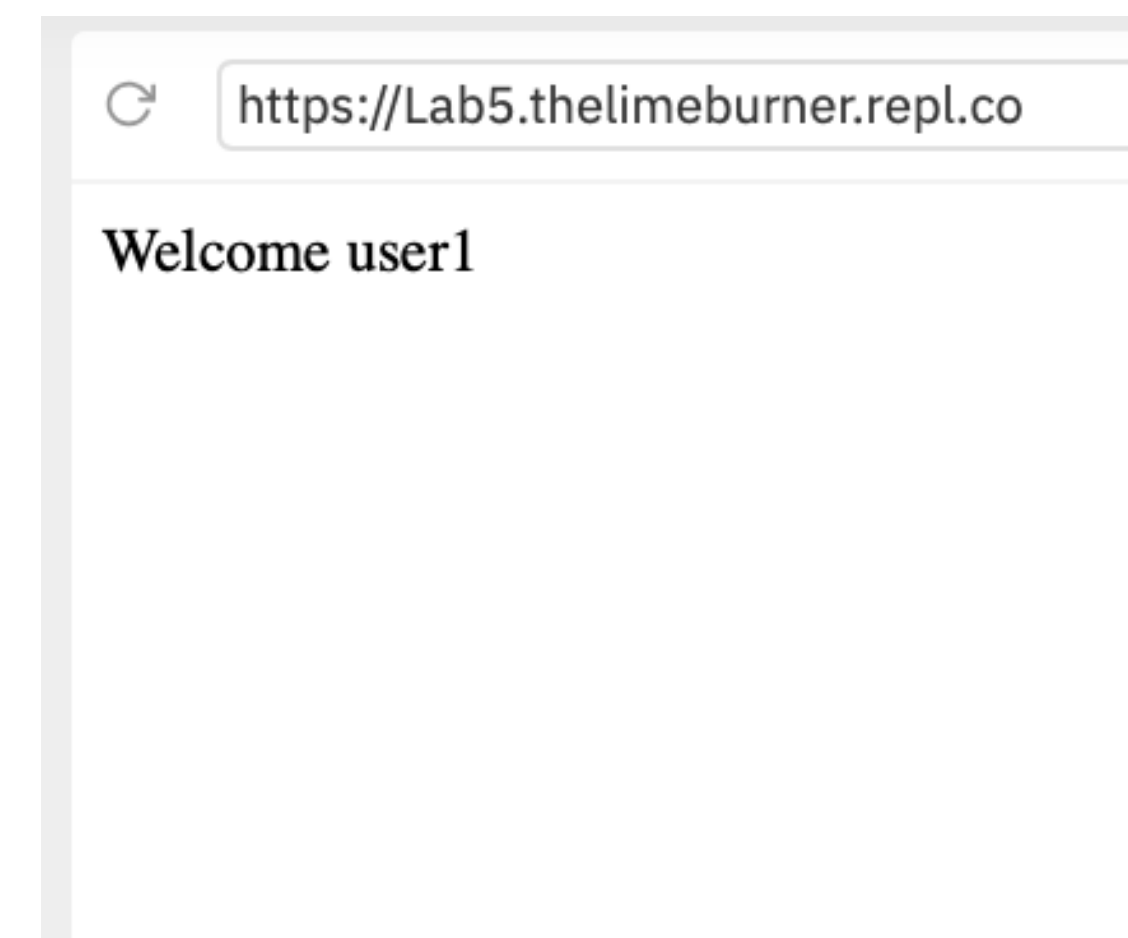
# Activity 1

<https://repl.it/@thelimeburner/Lab5-template#main.py>

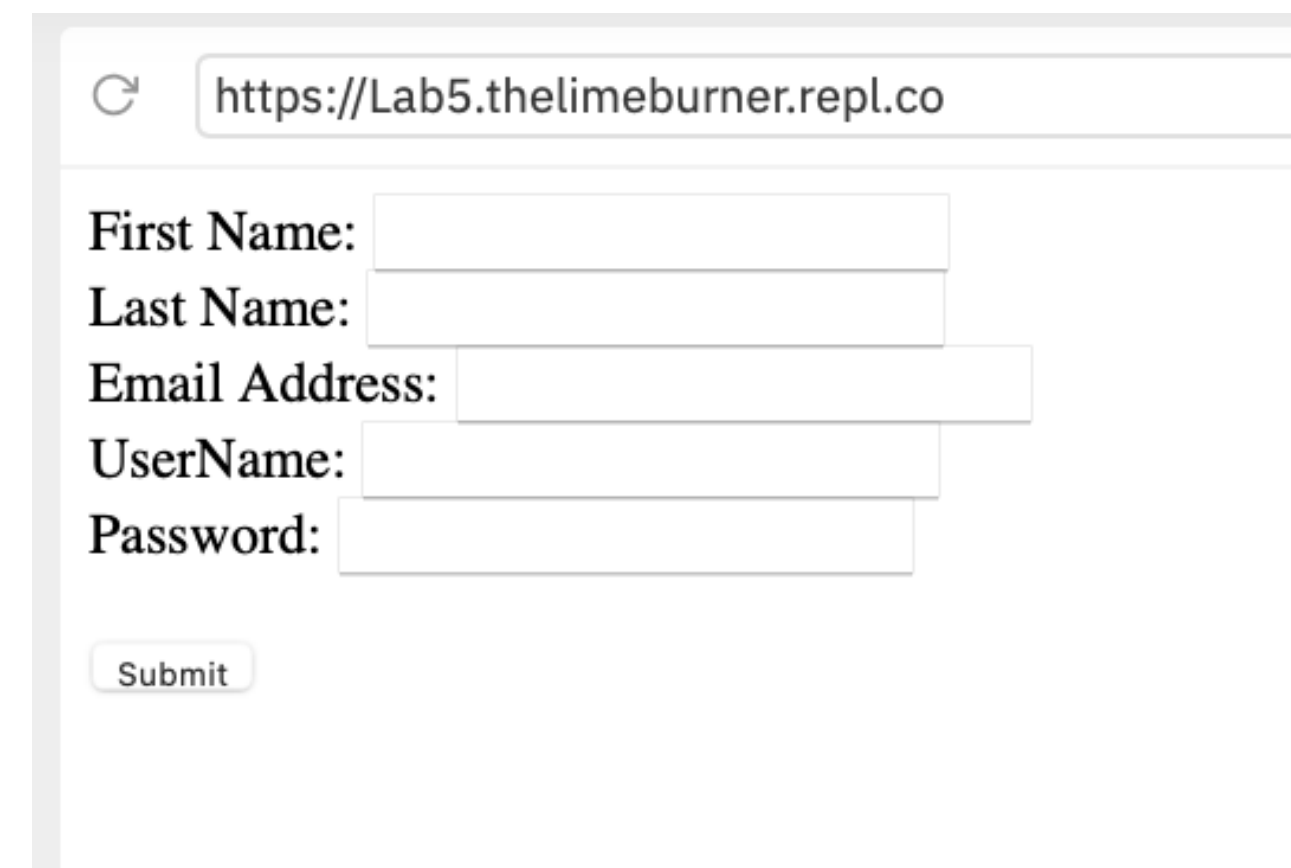
- Create a simple authentication form.
- Routes are predefined for you and the database is already created.
- You need to:
  - Present a form for logging in
  - Present a form for registering a user
  - Use sql queries to:
    - Check user authentication against correct value in database.
    - Store newly registered users in the database.



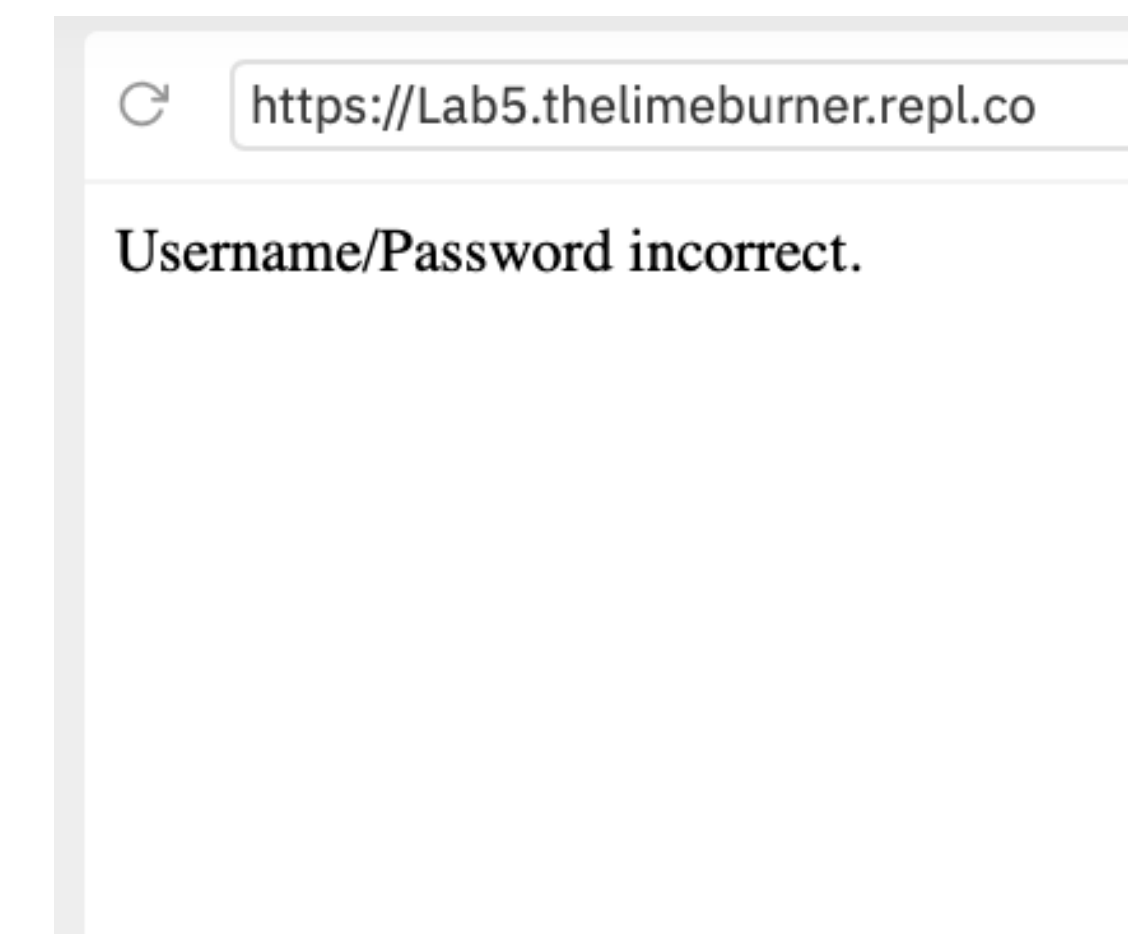
A screenshot of a web browser showing a login form. The address bar contains "https://Lab5.thelimeburner.repl.co". The form has two input fields: "UserName:" and "Password:". Below the fields is a "Submit" button. Underneath the button is a link that says "Register an Account" and a note: "\* Use sample account user1/password1 \*".



A screenshot of a web browser showing a successful login. The address bar contains "https://Lab5.thelimeburner.repl.co". The page displays the text "Welcome user1".



A screenshot of a web browser showing a registration form. The address bar contains "https://Lab5.thelimeburner.repl.co". The form has five input fields: "First Name:", "Last Name:", "Email Address:", "UserName:", and "Password:". Below the fields is a "Submit" button.



A screenshot of a web browser showing a failed login. The address bar contains "https://Lab5.thelimeburner.repl.co". The page displays the text "Username/Password incorrect."