11. Databases 2

And a LITTLE bit of JavaScript

Logistics ⚙️

- **Extra credit assignment posted:**
  - Complete and give feedback on job readiness modules on iCollege to replace your lowest HW grade with a 6/5.
  - If your lowest HW grade is a 4 or higher, I'll replace it with a 7/5.
- HW10 is out, and you have **two** weeks to complete it.
  - There will still be homework in future weeks, but it won't be coding (at least not in the same way)
- How was the Showcase??

Agenda 📆

1. Recap
2. DB tips
3. Flask tips
4. JavaScript???
5. In-class exercise: login

Recap: HTTP methods

What set of HTTP methods does @app.route() accept by default?

1) All of them
2) POST only
3) GET only
4) GET and POST

Recap: HTTP methods

What set of HTTP methods does @app.route() accept by default?

1) All of them
2) POST only
3) GET only
4) GET and POST

Recap: SQLAlchemy

With SQLAlchemy, which of the following do we define Python classes to represent:

1) A database
2) A table
3) A row in a table
4) A column in a table

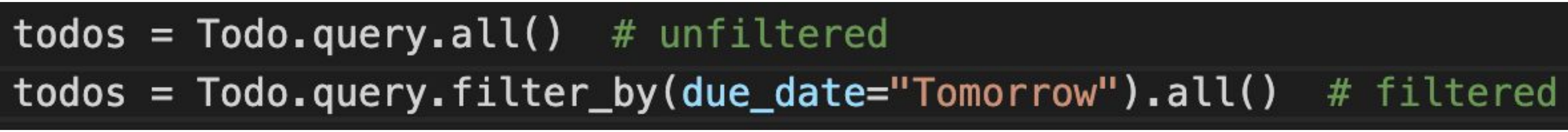
Recap: SQLAlchemy

With SQLAlchemy, which of the following do we define Python classes to represent:

1) A database
2) A table
3) A row in a table
4) A column in a table

One more thing: DB filtering

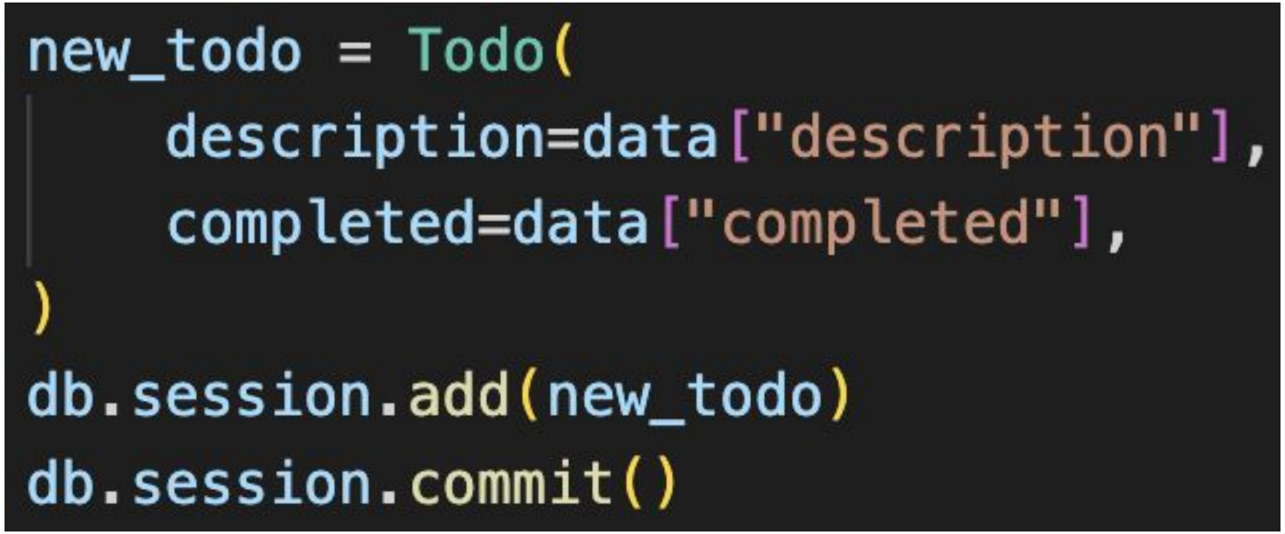
- We didn’t do this in the demo last time
- Most of the reason you’d want a relational DB is to be able to query for objects with specific traits!



```
todos = Todo.query.all()  # unfiltered
todos = Todo.query.filter_by(due_date="Tomorrow").all()  # filtered
```

One more one more thing: removing from a DB

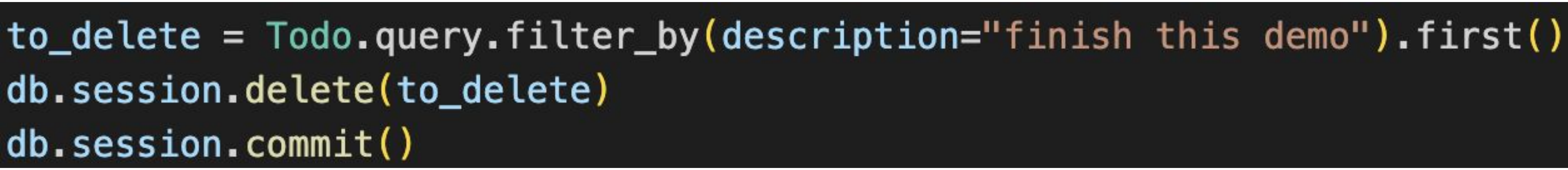
Recall that you can add to a DB like so:



```
new_todo = Todo(
    description=data["description"],
    completed=data["completed"],
)
db.session.add(new_todo)
db.session.commit()
```

One more one more thing: removing from a DB

You can also delete things from a DB - but you have to query for the object you want first!



```
to_delete = Todo.query.filter_by(description="finish this demo").first()
db.session.delete(to_delete)
db.session.commit()
```

One more one more one more thing: debugging your DB

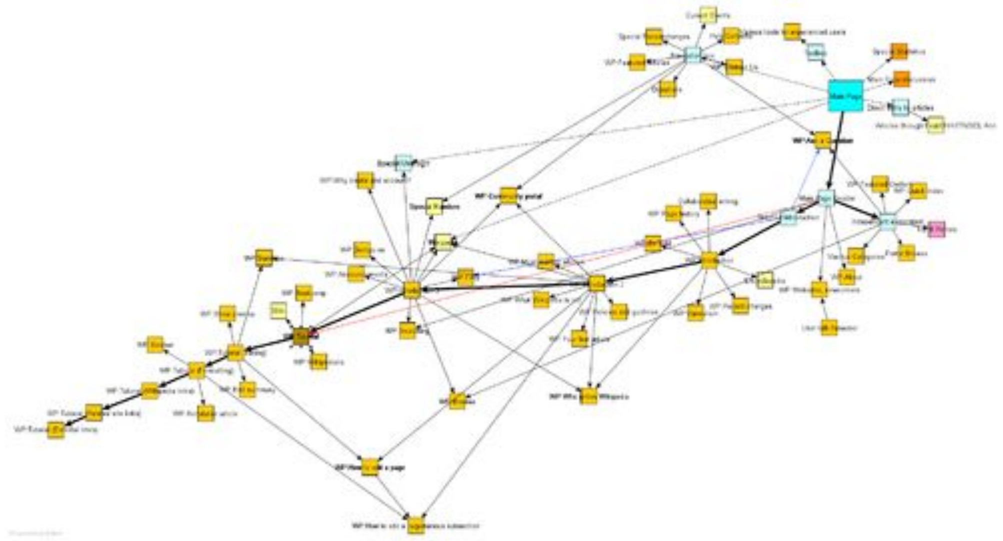
- Run `sqlite3` or `sqlite3.exe` followed by the name of your database from the command line.
- Use `.tables` to list all your tables and `.schema <table>` to see the schema for a table
- You can execute SQL statements in this shell (but don’t forget the semicolon at the end of a command!)
- Run `DROP TABLE <table_name>;` to delete your table.

*****You will have to do this if you ever want to change the schema of a table (add or delete a column) that has already been created!*****

Flask Tips

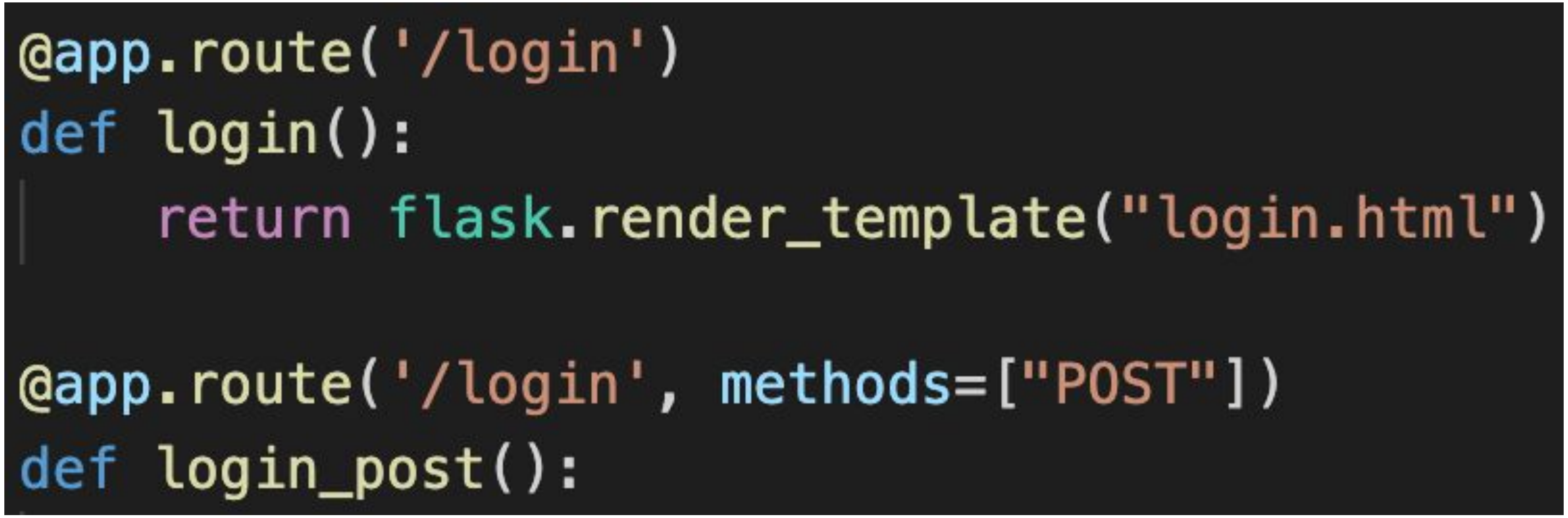
Multiple pages 📑

- So far our apps have all been single-page
- With frameworks like React, this is a perfectly viable way of doing web development
- But most modern websites still have multiple endpoints.



Multiple pages 📑

- Use @app.route()
- Design pattern: multiple endpoints for multiple methods?



```
@app.route('/login')
def login():
    return flask.render_template("login.html")

@app.route('/login', methods=["POST"])
def login_post():
```

What can a Flask app route return?

What can a Flask app route return?

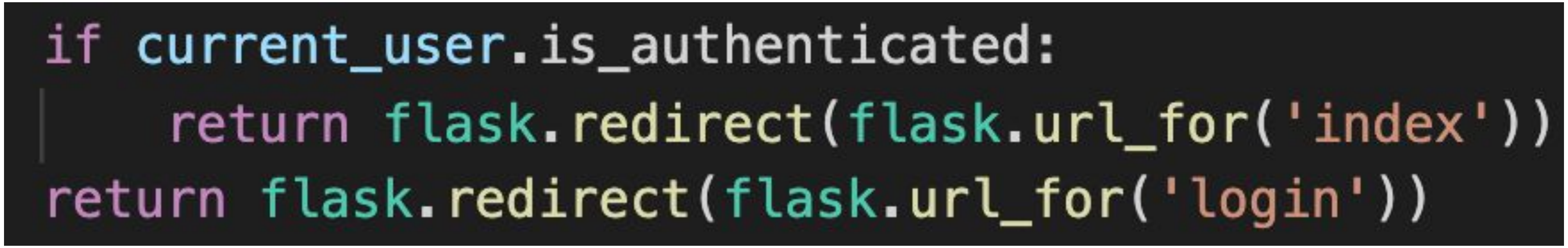
- It can send raw data (like JSON)!
- It can send us to a different page!

Redirects ↪

- We can send the user to another page after receiving a request
- When would we want to do this?

Redirects ↪

- flask.redirect to the rescue!

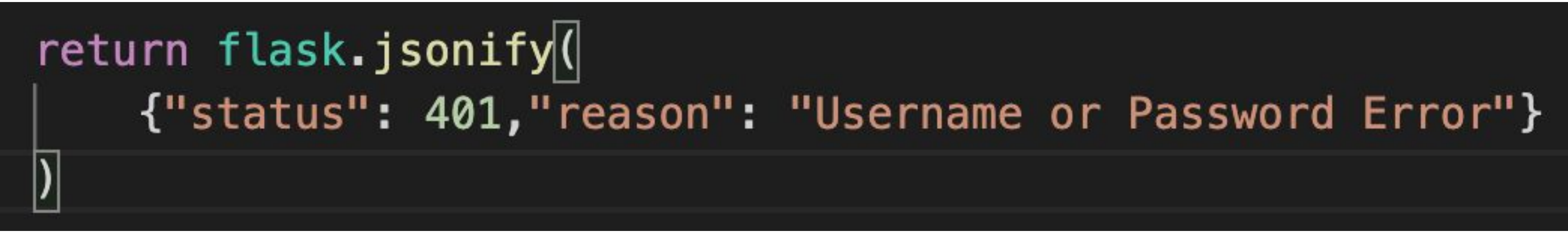


```python
if current_user.is_authenticated:
    return flask.redirect(flask.url_for('index'))
return flask.redirect(flask.url_for('login'))
```

JSON responses?

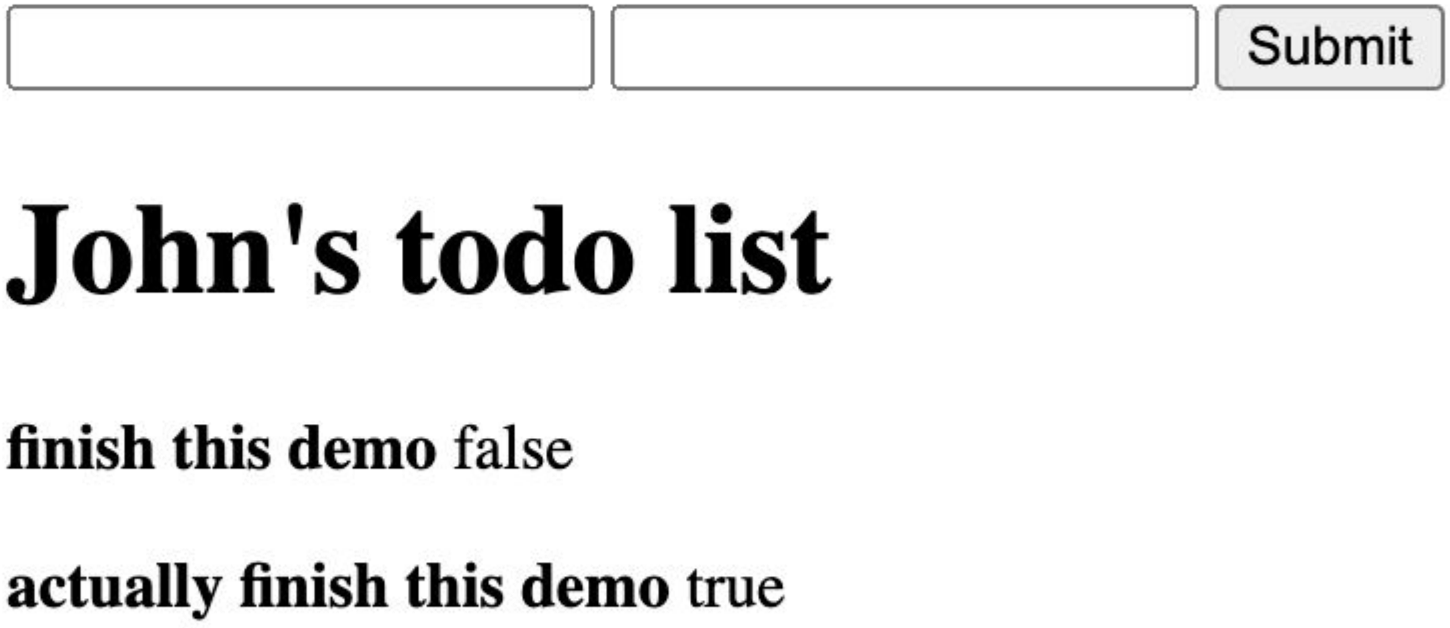
- What if you wanted to make a better Google Books API? Or your own REST API?
- Loading `api.nytimes.com` endpoints in your browser results in JSON!

JSON responses?



```
return flask.jsonify(
    {"status": 401,"reason": "Username or Password Error"}
)
```

Interlude: Let’s roast last class’s demo?

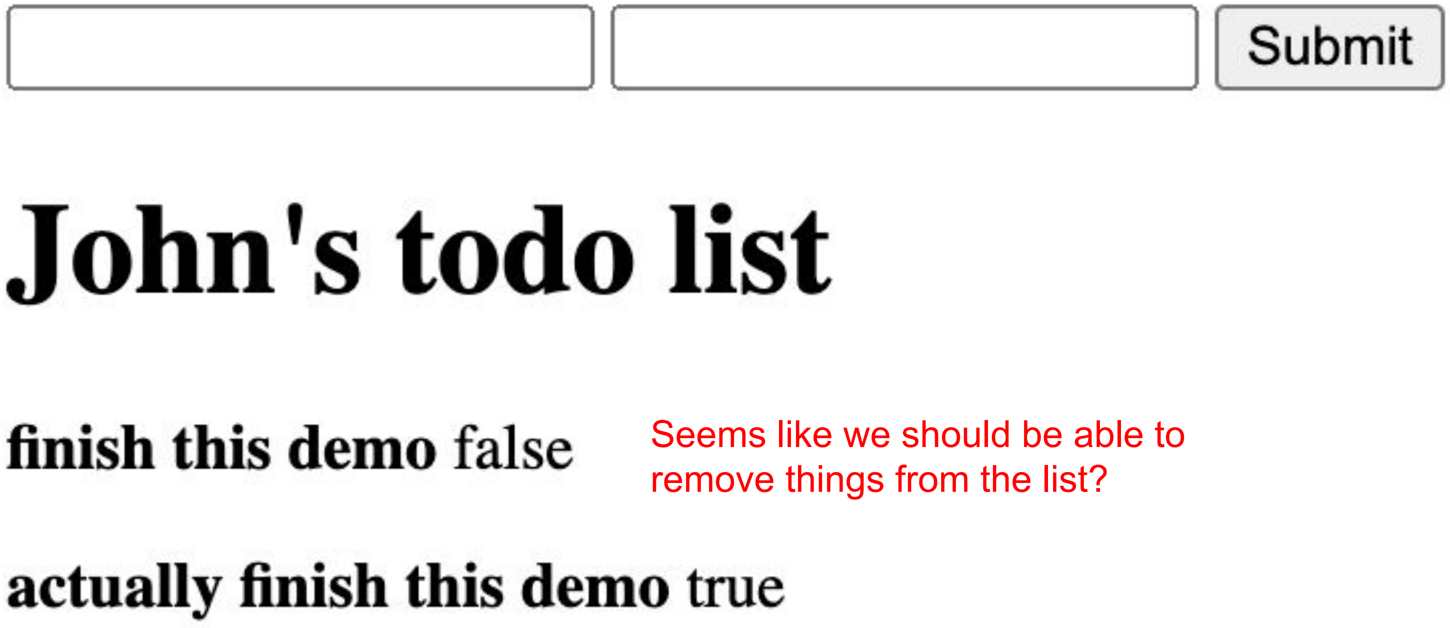


John's todo list

finish this demo false

actually finish this demo true

Interlude: Let’s roast last class’s demo?



John's todo list

finish this demo false

actually finish this demo true

Seems like we should be able to remove things from the list?

Break

JavaScript??

- Is code that runs in the **browser**
- It’s technically a static resource, just like CSS!
- Provides the cleanest way for us to talk to the server from the frontend.

JavaScript??

- We’re just going to focus on two commands:
- fetch() is the equivalent of Python’s requests.get (or requests.post)
- window.location = <page route> results in loading that page (or refreshing if it’s the current page)
- I’m not expecting you to be JS experts (or even close) after this lecture.

It’s demo time! ⏰

Finishing the Todo App

Your turn!

- Let’s to implement a simple login screen
- Users should be able to sign up with a username + password, and then only be redirected to the “main” page from the login page if they enter credentials that have been used to sign up in the past.
- What routes do we need? What HTML files do we need?
- (Hint: this part doesn’t involve “fetch” or any JS)

Later 👋

- Before next class, start HW10!
- That’s the last technical topic of the semester! From here on out we’re going to talk about the planning process and lifecycle of software projects!

Appendix: Resources - My class is **not** enough!

- SQLite CLI tool: https://sqlite.org/cli.html
- Flask-SQLAlchemy: https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/
- Fetch API: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- Google! Google! Google! (and Discord)