# Flask Tutorial — Build a Student Sign-In Website

## 1. Objective

This tutorial introduces the basic workflow of Flask web development. Students will learn to create a simple student sign-in website and understand how Flask handles routing, templates, forms, and session management.

## 2. Learning Outcomes

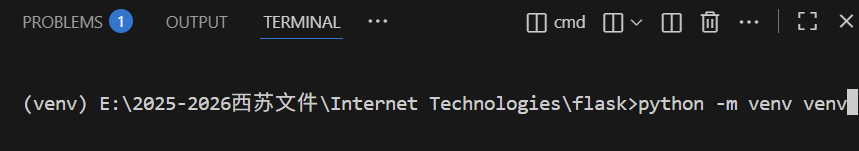
- Install and configure the Flask development environment.  
- Create and run a basic Flask web application.  
- Use the Jinja2 template engine to design dynamic web pages and handle user input.

- Utilize session technology to manage user state and render dynamic content.  
- Understand the basics of data persistence and production environment deployment.

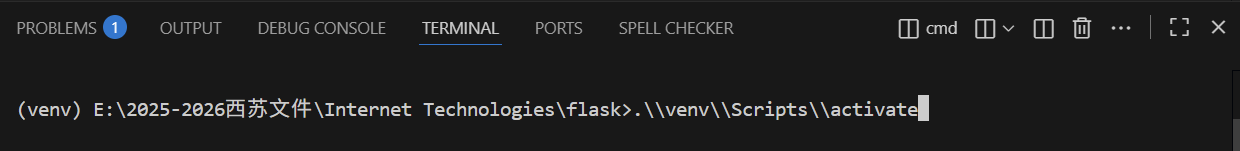
## 3. Environment Setup

Python 3.8 or higher is required. Install Flask using the command below:

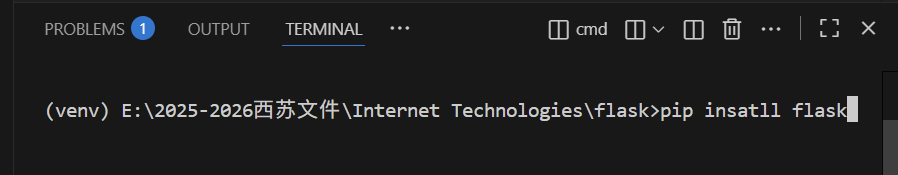
3.1 Open the VSCode terminal or press the shortcut key Ctrl + `, then enter the following command in the terminal to create a virtual environment.

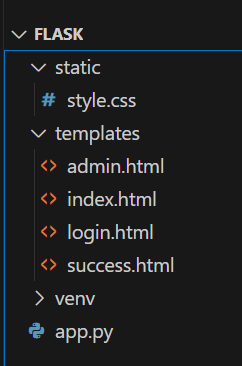


3.2 Activate the virtual environment: Taking Windows as an example



3.3 Install flask: After ensuring the virtual environment is activated, enter the following command in the terminal to install flask.



3.4 Recommended folder structure:  


## Core Components

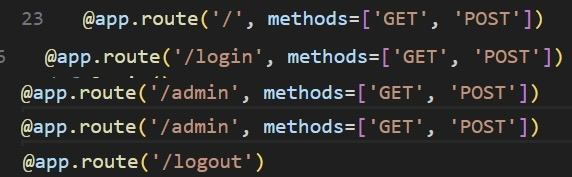
4.1 Routing (Static / Dynamic):

@app.route('/about')

**Static routing**: Fixed URL — Fixed view function, suitable for pages like "About/Help".

@app.route('/user/')

**Dynamic routing**: Often used to receive path variables

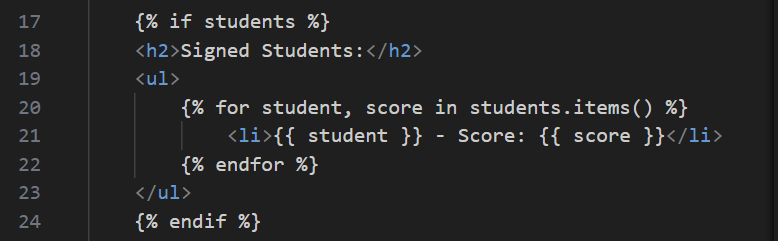


4.2 Jinja2 Templates: Inheritance and Loop

templet：{% for item in items %} {{ item }} {% endfor %}

Jinja2 template syntax: {% for %} for loops, {{ item }} outputs variable values; rendering occurs on the server side, returning pure HTML.

Templates allow embedding Python variables into HTML, achieving separation of business logic and presentation layer.

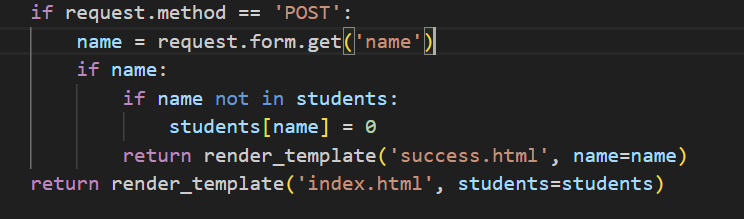


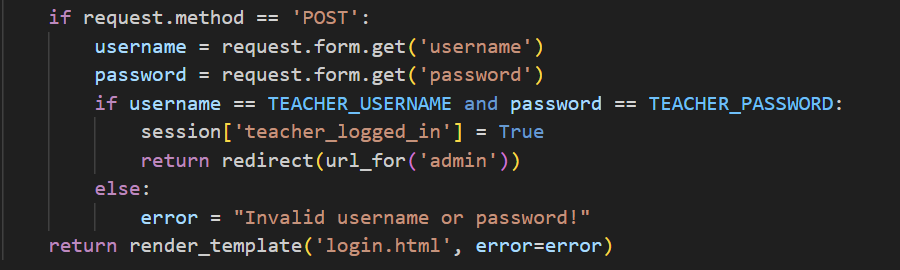
Render the template in the view function:

return render\_template('list.html', items=my\_list)

Read list.html from the templates directory, inject items=my\_list; achieve the separation of "logic (Python) / presentation (HTML)".

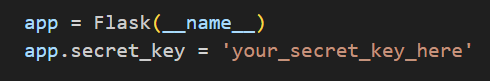
4.3 Receiving browser data



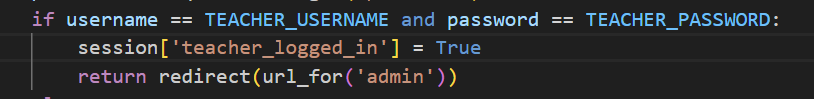


This primarily utilizes request.form for scenarios such as reading student and teacher data login submission forms.

4.4 Redirects & Session (the three lines from PPT)



Enable session signing (and some security features); production environments must use complex random values.



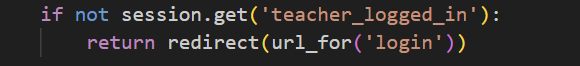
Write the login username into the session to achieve "cross-request" login state persistence.

Login successful, writing to teacher\_logged\_in

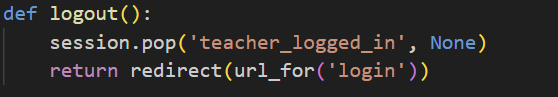


PRG (Post/Redirect/Get) Pattern: Redirect after form submission to prevent duplicate submissions caused by refreshing

Login successful redirect



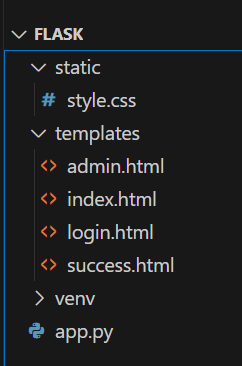
Redirect when accessing the management page without logging in.



Redirect after login

## **5. Practical Code**

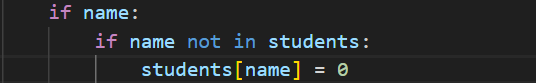
5.1 Project directory structure



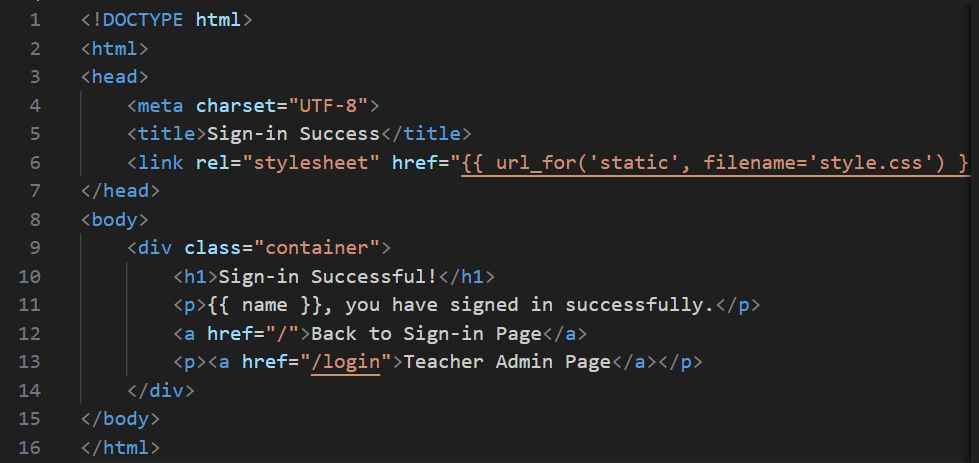
5.2 Student sign-in page core logic

Initialize the first occurrence of a name to 0 points to avoid errors (missing key) when adding/subtracting points later.

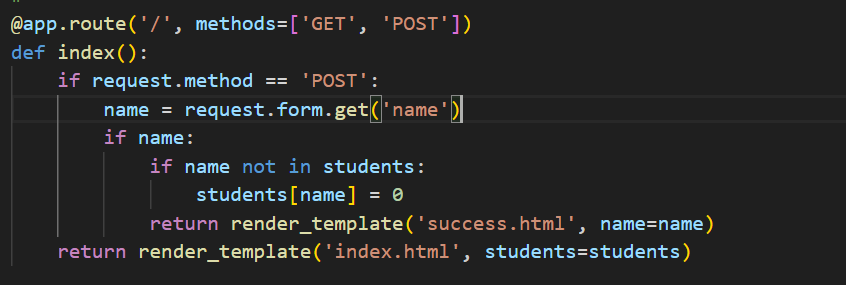
Students is an in-memory dictionary for demonstration purposes: data will be lost upon restart.



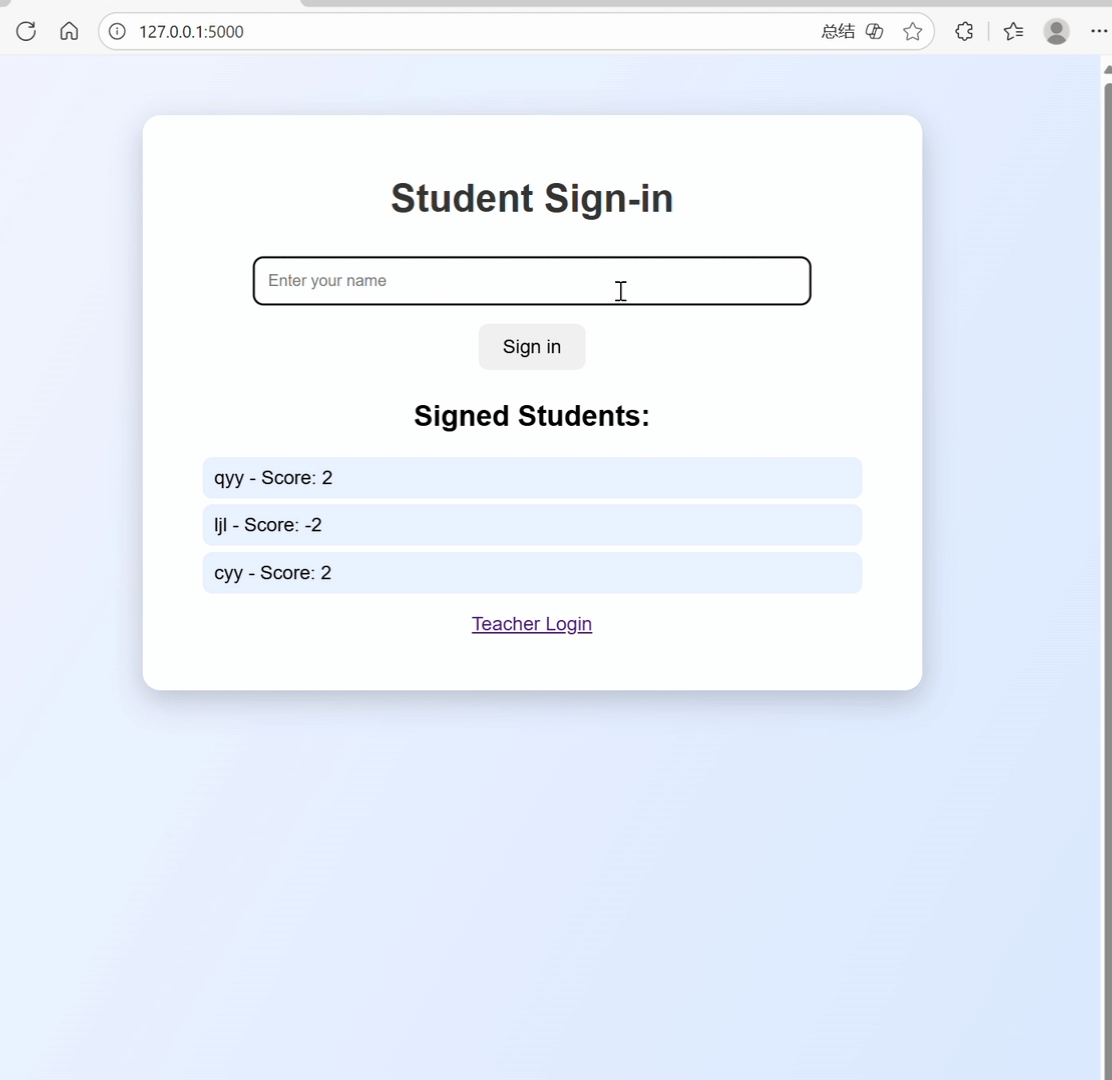
success.html



app.py



Functionality demo(Student Edition)

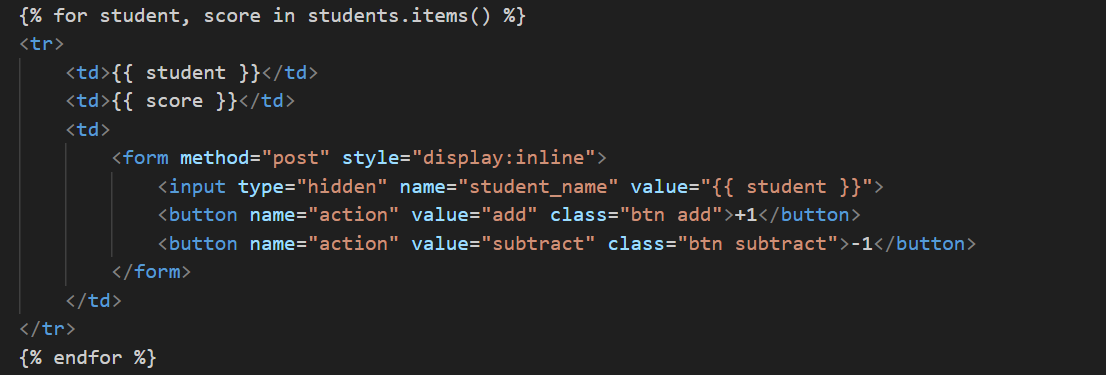


5.3 Teacher portal

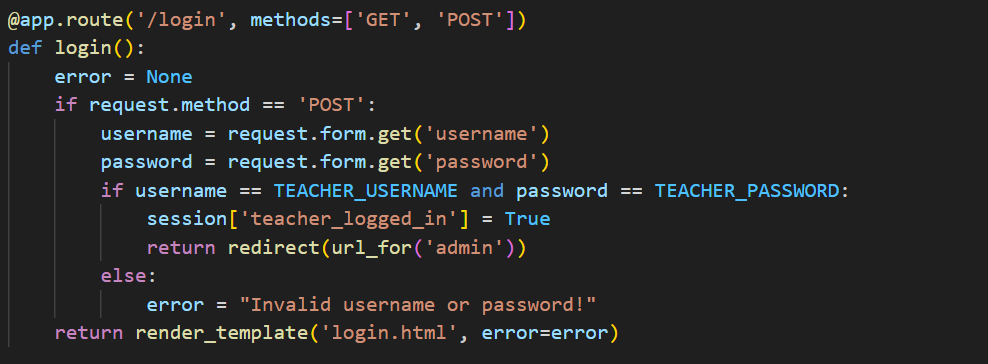
Iterate students dict in a table and add an action button for each row

Render students.items() in a table row by row; place "+1/-1" buttons in each row (hidden fields carry the name).

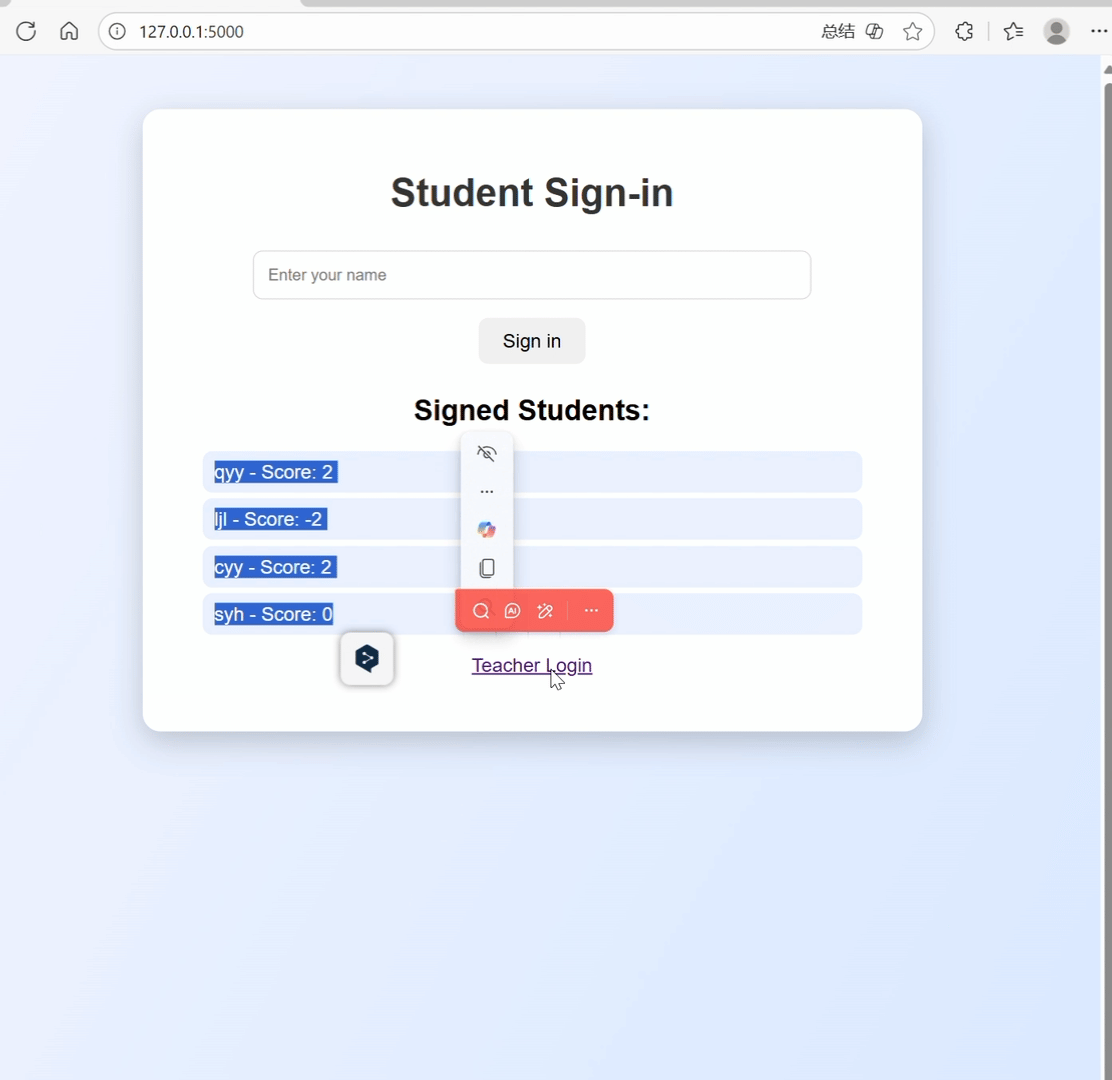
Upon receiving a POST request, the backend adjusts the scores and redirects back to this page to display the latest scores.



app.py



Feature Demo (Teacher Edition)



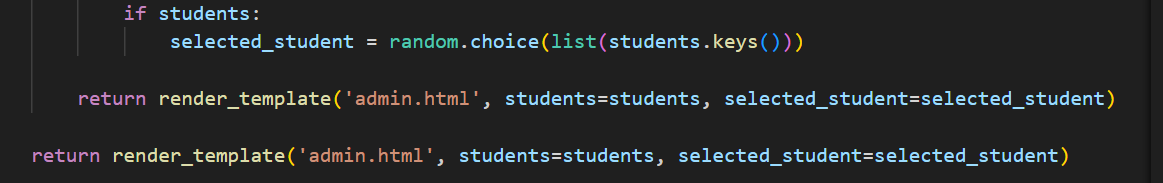
5.4 Bonus/Penalty & Random Roll Call

Trigger button：flask/templates/admin.html

Form name= random\_draw (Bottom random button)

## 

Random selection logic: flask/app.py displays the selected result after execution



## 6. Deployment & Scaling

## 6.1 Production environment startup commands

Flask's built-in development server is not suitable for production. We can use Waitress or Gunicorn.

**Deploy command：**

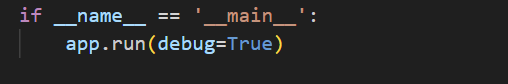
pip install waitress

waitress-serve --port=5000 app:app

pip install gunicorn

gunicorn --bind 0.0.0.0:5000 app:app

6.2 Development & Security



* Debug Mode: Set ‘debug=True’ during development for hot reloading and detailed error pages.
* Secret Key Management: The ‘SECRET\_KEY’ is crucial for session security. It must be a strong random value and must never be committed to a public repository.

## 7. Running the Application

1. Run the following command in your terminal:  
python app.py  
  
2. Then open http://127.0.0.1:5000 in your browser.

3. Try signing in with a name, and observe the attendance list and admin functions.

## 8. Discussion and Extensions

This exercise demonstrates how Flask integrates Python with web interfaces. Possible improvements include adding database support with SQLite, login features using Flask-Login, and asynchronous updates via AJAX.

## 9. Conclusion

Through this tutorial, learners can understand the foundation of Flask web development and gain confidence to expand into more complex web applications.

## 10. References

• Albesher, L., & Alfayez, R. (2024). An Observational Study on Flask Web Framework Questions on Stack Overflow (SO). IET Software.

• Bonney, M., de Angelis, M., Dal Borgo, M., Andrade, L., Beregi, S., Jamia, N., & Wagg, D. (2022). Development of a Digital Twin Operational Platform Using Python Flask. Data-Centric Engineering.

• Liang, Z., Liang, Z., Zheng, Y., Liang, B., & Zheng, L. (2021). Data Analysis and Visualization Platform Design for Batteries Using Flask-Based Python Web Service. World Electric Vehicle Journal.

• Zhang, D., & You, S. (2020). iFlask: Isolate Flask Security System from Dangerous Execution Environment by Using ARM TrustZone. Future Generation Computer Systems.

• Guirales, S., Machado, D., Schneider, A., & Janies, D. (2020). FLAVi-Web: A Web Annotator for Viral Genomes of Flaviviridae with Flask. The FASEB Journal.

• Wang, Z., Xiang, H., & Zhang, L. (2024). Design of a Classroom Big Data Analysis System Based on Open Source Technology. ICEIT.

• Li, Z. (2023). The Construction of Online Learning System Based on Flask.

• DeCausemaker, R., & Jacobs, S. (2015).

• Landschaft, O., & Wishkerman, A. (2024). FlaskLED: An Additive Manufacturing Approach for Low-Cost Bioreactors. Journal of Applied Phycology.