Python Flask Blog Tutorial

Contents

Python Flask Blog Tutorial 1
Step 1: Project Setup and Installation..1
Step 2: Creating the Main Application 2
Step 3: Building the HTML Templates 4
Step 5: Run Your Blog! 7
Assignment Submission..8

Welcome! In this tutorial, we'll build a simple but functional blog from scratch. We'll use:

- **Python**: Our programming language.
- **Flask**: A lightweight "microframework" for building web applications in Python. It's simple and flexible, making it perfect for beginners.
- **SQLAlchemy**: An Object-Relational Mapper (ORM). Instead of writing raw SQL commands, SQLAlchemy allows us to interact with our database using simple Python classes and objects. This makes our code cleaner, more readable, and easier to maintain.

By the end, you'll have a web app where you can view and create new blog posts. Let's get started!

Step 1: Project Setup and Installation

Let's get our project folder and virtual environment set up. Using a **virtual environment** is a best practice because it keeps the dependencies for this project separate from others on your system.

1. Create a project folder named **blog_tutorial**
2. Create and Activate a Virtual Environment:
    - On macOS/Linux: Open a terminal in the project directory.

```
python -m venv .venv
source .venv/bin/activate
```

- On Windows: Open a command prompt in the project directory.

```
python -m venv .venv
.venv\Scripts\activate
```

You'll know it's working when you see (.venv) at the beginning of your command prompt.

3. **Install Flask**: With the virtual environment active, install Flask using pip.

```
pip install Flask Flask-SQLAlchemy
```

Your project folder should now contain a **.venv** directory.

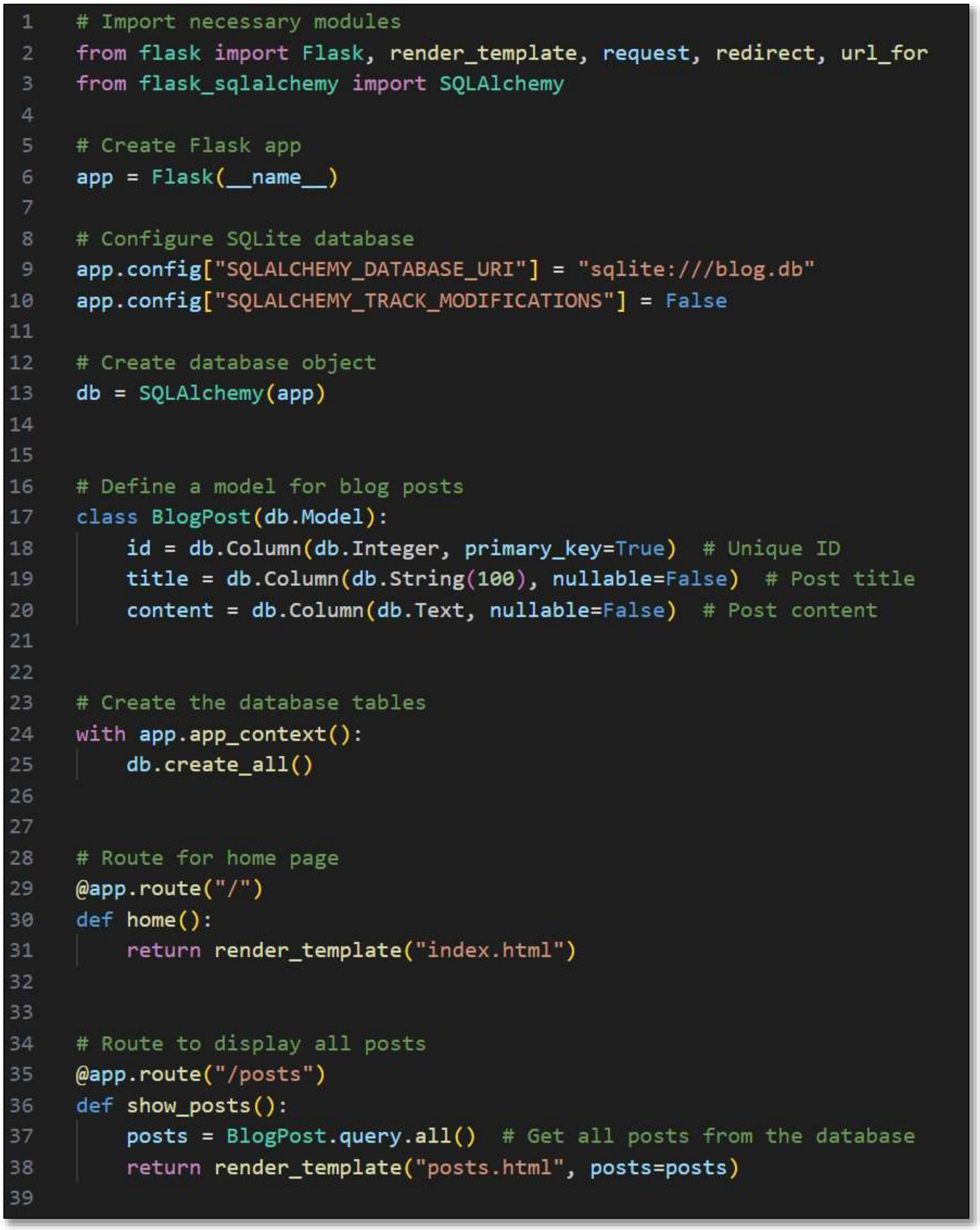
Step 2: Creating the Main Application

Create the project folders as shown:

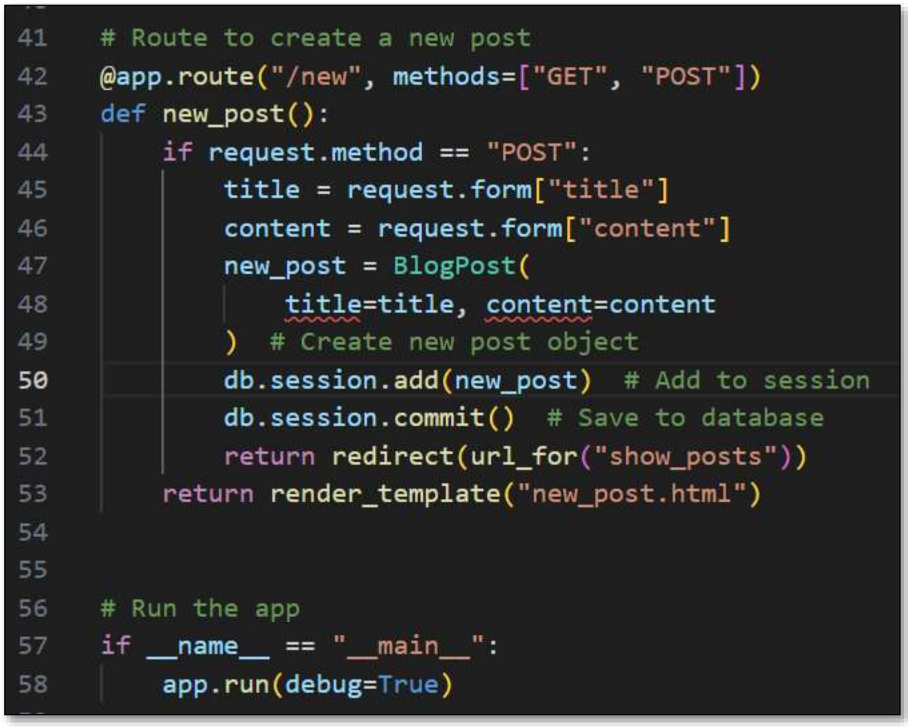
```
blog_tutorial/
 ├── app.py
 ├── static/css/
        └── style.css
 └── templates/
     ├── base.html
     ├── index.html
     ├── new_post.html
     └── posts.html
```

It's time to write the core of our application. We'll create a single Python file, **app.py**, that will define our database structure, handle web page requests, and contain all of our application's logic.

Create a new file named **app.py** and add the following code. The comments walk you through what each section does.



```python
# Import necessary modules
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

# Create Flask app
app = Flask(__name__)

# Configure SQLite database
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///blog.db"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False

# Create database object
db = SQLAlchemy(app)


# Define a model for blog posts
class BlogPost(db.Model):
    id = db.Column(db.Integer, primary_key=True)  # Unique ID
    title = db.Column(db.String(100), nullable=False)  # Post title
    content = db.Column(db.Text, nullable=False)  # Post content


# Create the database tables
with app.app_context():
    db.create_all()


# Route for home page
@app.route("/")
def home():
    return render_template("index.html")


# Route to display all posts
@app.route("/posts")
def show_posts():
    posts = BlogPost.query.all()  # Get all posts from the database
    return render_template("posts.html", posts=posts)

```

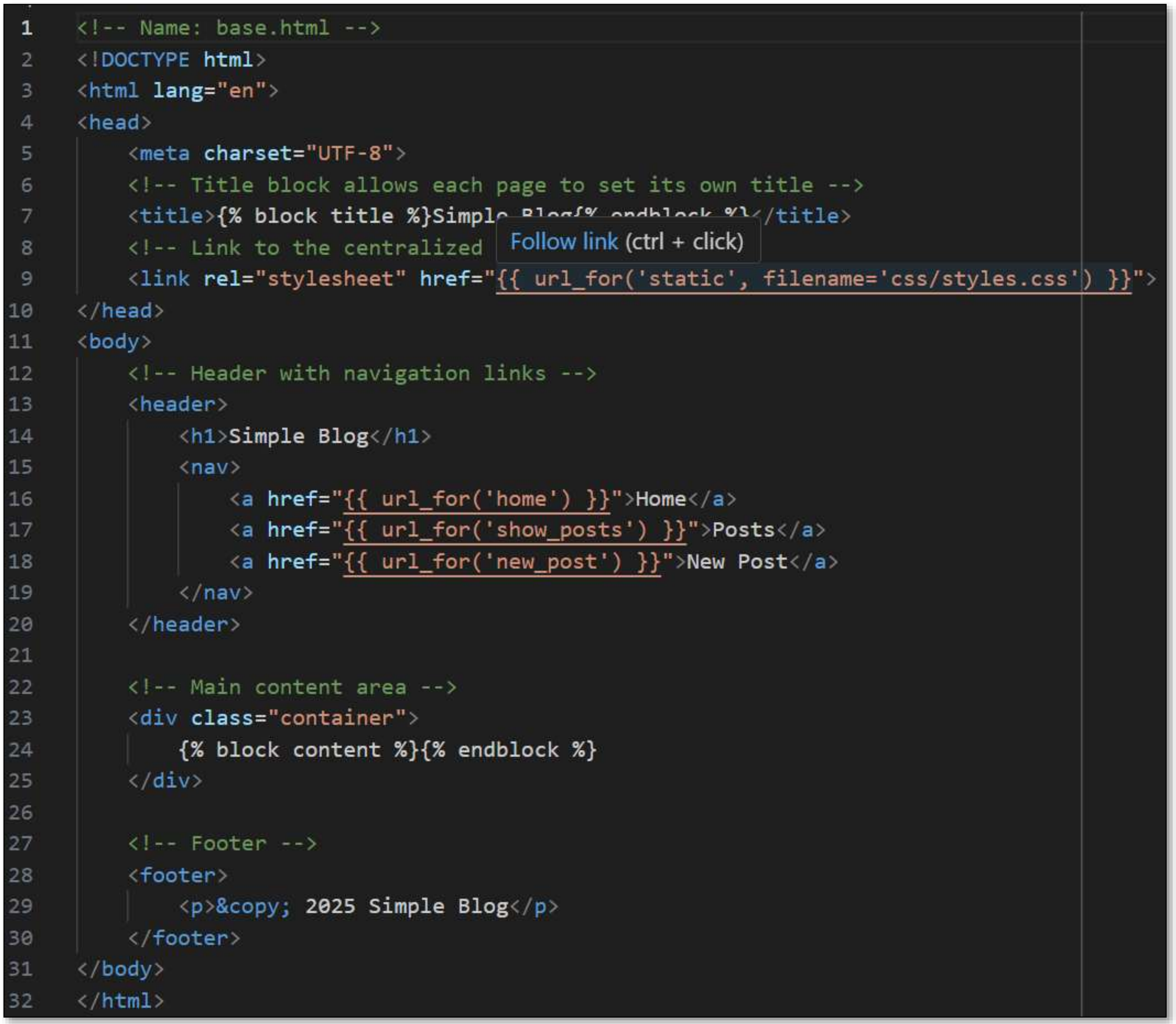


```
# Route to create a new post
@app.route("/new", methods=["GET", "POST"])
def new_post():
    if request.method == "POST":
        title = request.form["title"]
        content = request.form["content"]
        new_post = BlogPost(
            title=title, content=content
        )  # Create new post object
        db.session.add(new_post)  # Add to session
        db.session.commit()  # Save to database
        return redirect(url_for("show_posts"))
    return render_template("new_post.html")


# Run the app
if __name__ == "__main__":
    app.run(debug=True)
```

Step 3: Building the HTML Templates

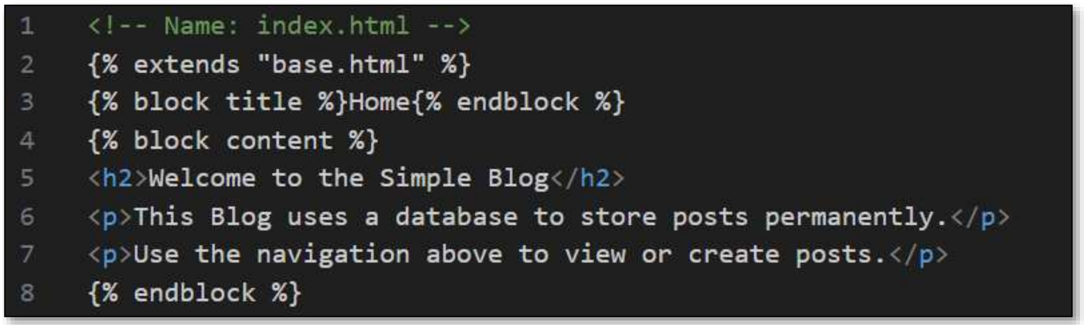
Our Flask application needs HTML files to show users in their browsers. Flask automatically looks for these files in a folder named templates.

1. **Create the Base Template (base.html)**: A base template contains the common HTML structure (like the navigation bar and basic styling) that other pages will inherit.
2. Create the file **templates/base.html**



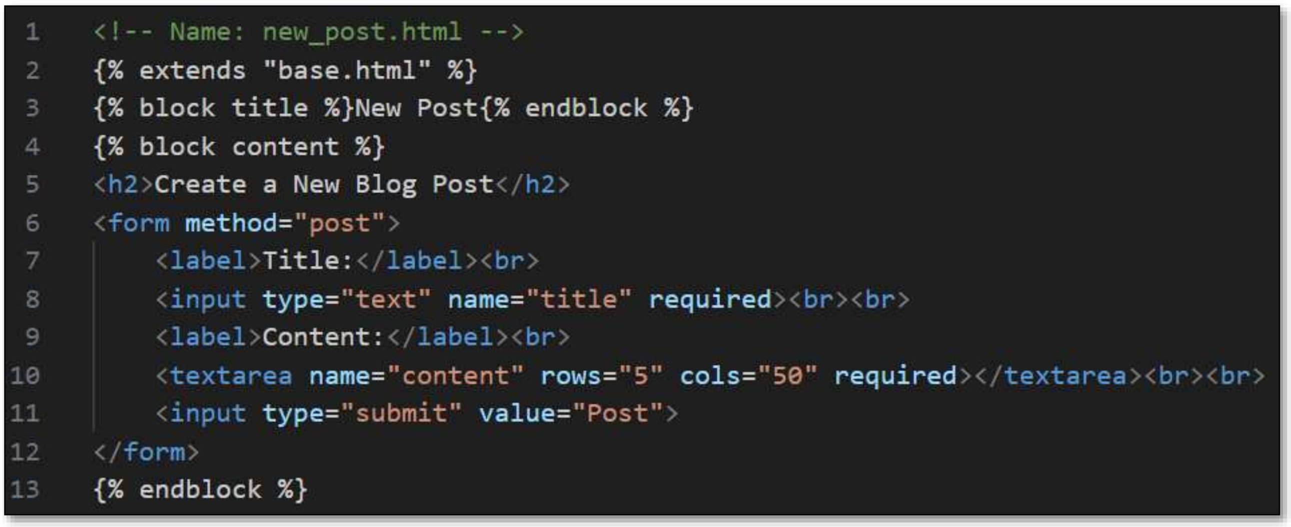
```html
<!-- Name: base.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <!-- Title block allows each page to set its own title -->
    <title>{% block title %}Simple Blog{% endblock %}</title>
    <!-- Link to the centralized
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
    <!-- Header with navigation links -->
    <header>
        <h1>Simple Blog</h1>
        <nav>
            <a href="{{ url_for('home') }}">Home</a>
            <a href="{{ url_for('show_posts') }}">Posts</a>
            <a href="{{ url_for('new_post') }}">New Post</a>
        </nav>
    </header>

    <!-- Main content area -->
    <div class="container">
        {% block content %}{% endblock %}
    </div>

    <!-- Footer -->
    <footer>
        <p>&copy; 2025 Simple Blog</p>
    </footer>
</body>
</html>
```

3. Create **templates/index.html**



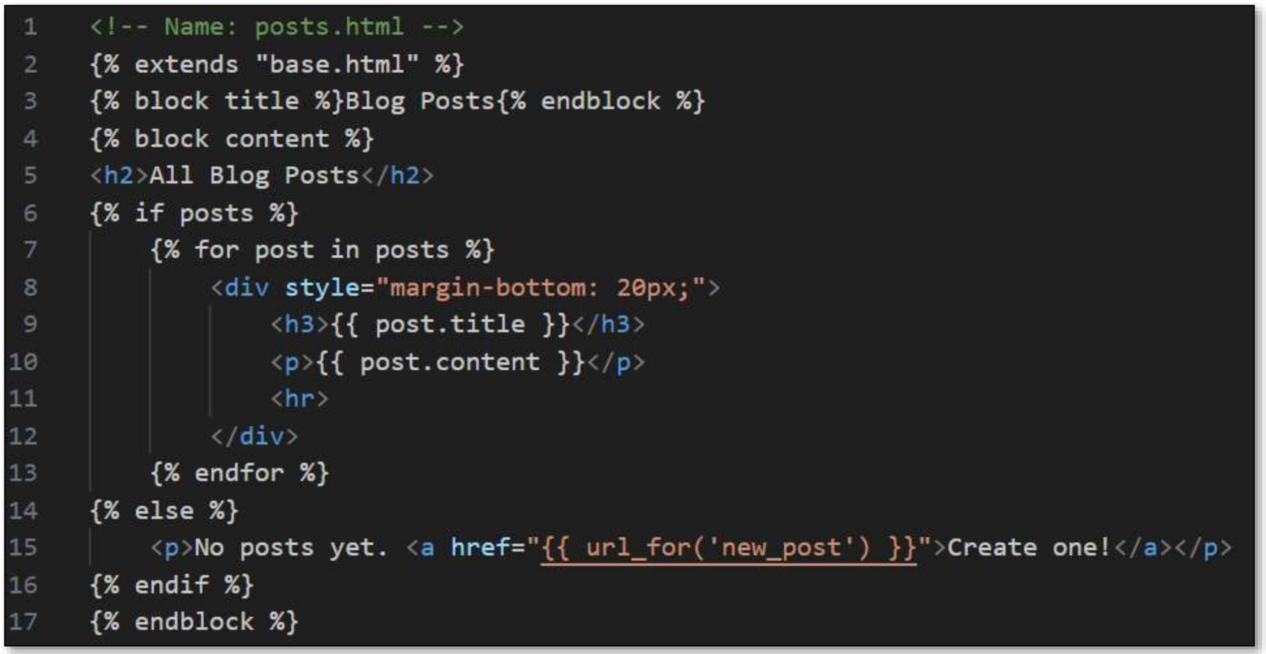
```html
<!-- Name: index.html -->
{% extends "base.html" %}
{% block title %}Home{% endblock %}
{% block content %}
<h2>Welcome to the Simple Blog</h2>
<p>This Blog uses a database to store posts permanently.</p>
<p>Use the navigation above to view or create posts.</p>
{% endblock %}
```

4. Create **templates/new_post.html**



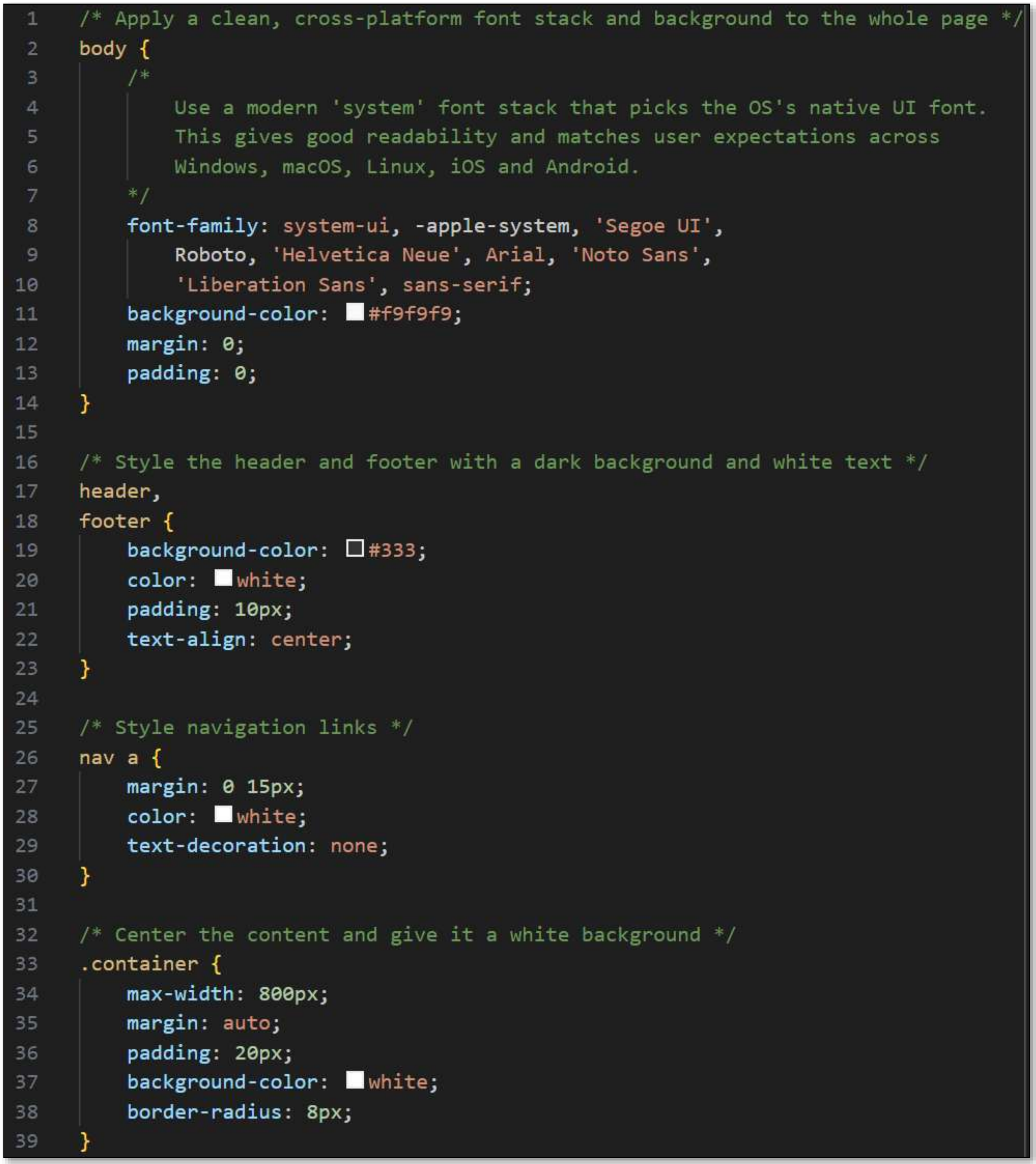
```html
<!-- Name: new_post.html -->
{% extends "base.html" %}
{% block title %}New Post{% endblock %}
{% block content %}
<h2>Create a New Blog Post</h2>
<form method="post">
    <label>Title:</label><br>
    <input type="text" name="title" required><br><br>
    <label>Content:</label><br>
    <textarea name="content" rows="5" cols="50" required></textarea><br><br>
    <input type="submit" value="Post">
</form>
{% endblock %}
```

5. Create **templates/posts.html**



```html
<!-- Name: posts.html -->
{% extends "base.html" %}
{% block title %}Blog Posts{% endblock %}
{% block content %}
<h2>All Blog Posts</h2>
{% if posts %}
    {% for post in posts %}
        <div style="margin-bottom: 20px;">
            <h3>{{ post.title }}</h3>
            <p>{{ post.content }}</p>
            <hr>
        </div>
    {% endfor %}
{% else %}
    <p>No posts yet. <a href="{{ url_for('new_post') }}">Create one!</a></p>
{% endif %}
{% endblock %}
```

6. Create **static/css/styles.css** This file contains the styling for the html pages.



```css
/* Apply a clean, cross-platform font stack and background to the whole page */
body {
    /*
        Use a modern 'system' font stack that picks the OS's native UI font.
        This gives good readability and matches user expectations across
        Windows, macOS, Linux, iOS and Android.
    */
    font-family: system-ui, -apple-system, 'Segoe UI',
        Roboto, 'Helvetica Neue', Arial, 'Noto Sans',
        'Liberation Sans', sans-serif;
    background-color: #f9f9f9;
    margin: 0;
    padding: 0;
}

/* Style the header and footer with a dark background and white text */
header,
footer {
    background-color: #333;
    color: white;
    padding: 10px;
    text-align: center;
}

/* Style navigation links */
nav a {
    margin: 0 15px;
    color: white;
    text-decoration: none;
}

/* Center the content and give it a white background */
.container {
    max-width: 800px;
    margin: auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
}
```

Step 5: Run Your Blog!

Everything is in place. Let's run the Flask development server to see our blog in action.

View Your Blog:

1. Open your web browser and navigate to **http://127.0.0.1:5000**.
2. You should see your blog's homepage. It will be empty at first.
3. Click "Create New Post," write your first entry, and hit submit.
4. Your new post will appear on the main page!

Assignment Submission

- Insert a screenshot showing a successful run of your program.
- Publish your project to your GitHub.
- Include a link to the GitHub repository.