CSCB20
Introduction to Databases and Web Application

All about Flask!

Dr. Purva R Gawde

Python Flask

- What is Flask?
  - Flask is an API of Python that allows to build web applications
  - Web application framework
    - Collection of modules and libraries that helps the developer to write applications without writing the low-level codes
  - Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.
- Why Flask?
  - Microframework
  - easier to learn

Flask

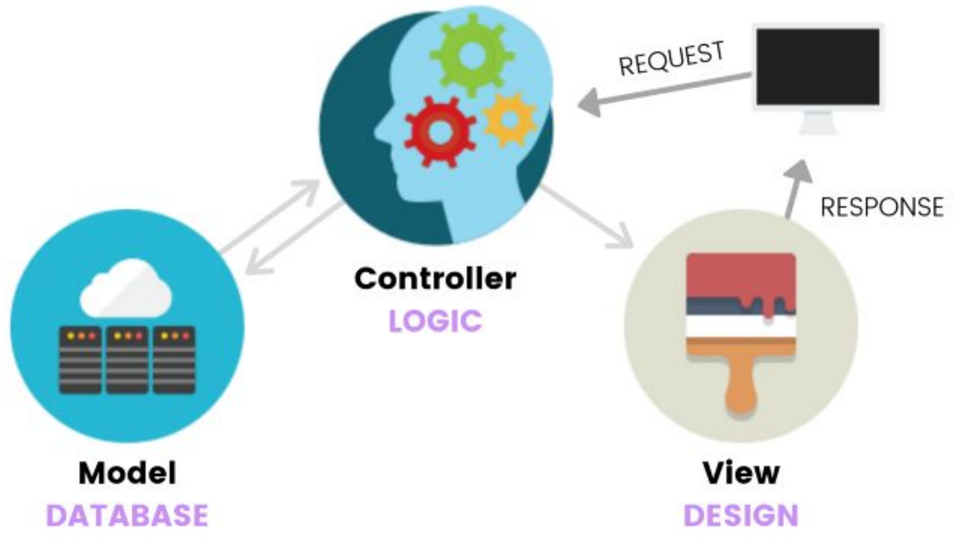
- Three dependencies:
  - Werkzeug: The routing, debugging, and Web Server Gateway Interface (WSGI) subsystems
  - Jinja2: Template engine
  - Click: command line integration
- No native support for:
  - accessing databases
  - validating web forms
  - authenticating users

Python Flask

- #1: What is Flask? Why Should You Care?
    - Flask is a lightweight and extensible Python web framework
- #2: Flask Structure
    - Instead of cramming all your code into one place, Flask helps you organize
        - (1) your logic,
        - (2) design, and
        - (3) database into separate files.

Python Flask Structure

- Logic:
  - 'main.py' imports the Flask module, creates a web server, creates an instance of the Flask class
- Design:
  - HTML files in templates folder
  - CSS in static folder
- Database:
  - SQLAlchemy supports a long list of database engines, including the SQLite (Grinberg).



Source: from Web Programming with Flask — Intro to Computer Science — Harvard's CS50 (2018)

What is HTTP and What Does it Have to do with Flask?

- HTTP (Hypertext Transfer Protocol) is the protocol for websites.
- The internet uses it to interact and communicate with computers and servers.
- When you enter website address, HTTP request is sent to server
- How is Flask involved?
    - We will write code that will take care of the server side processing.
- What is the role of Flask?
    - Flask lets us focus on what the users are requesting and what sort of response to give back.

Before coding!!

Download Flask
Create virtual Environment

Python Flask Installation

- Python Version:
- We recommend using the latest version of Python 3.
- Flask supports Python 3.5 and newer, Python 2.7, and PyPy.
- Flask:
    - https://flask.palletsprojects.com/en/3.0.x/

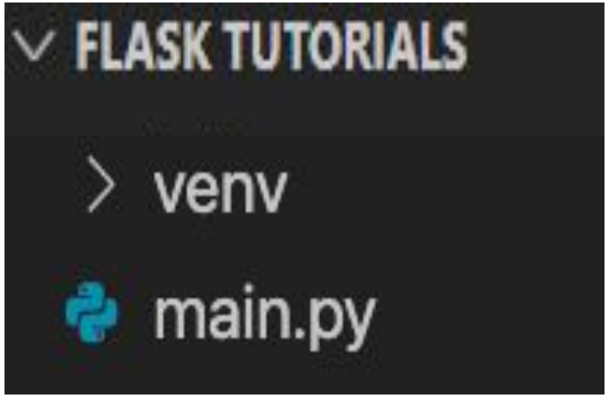
Virtual Environment

- Create an application directory for storing your code
- In this directory, install flask by first creating virtual environment.
- Why?
  - prevent package clutter and version conflicts in the system’s Python interpreter
  - ensures that applications have access only to the packages that they use
- Creating virtual environment with Python3
  - With Python 3, virtual environments are supported natively by the venv package that is part of the Python standard library
  - Inside the created directory, run the following command:
    - `python3 -m venv venv`

Activating Virtual Environment

- Make sure you are in the created directory:
    - Activation command for mac:
        - `source venv/bin/activate`
    - Activation for windows:
        - `venv\Scripts\activate`
    - How do we know we are using virtual environment?
        - `The command prompt includes the name of the environment:`
        - `(venv) $`
    - Install flask in the virtual environment
        - `Pip install Flask`

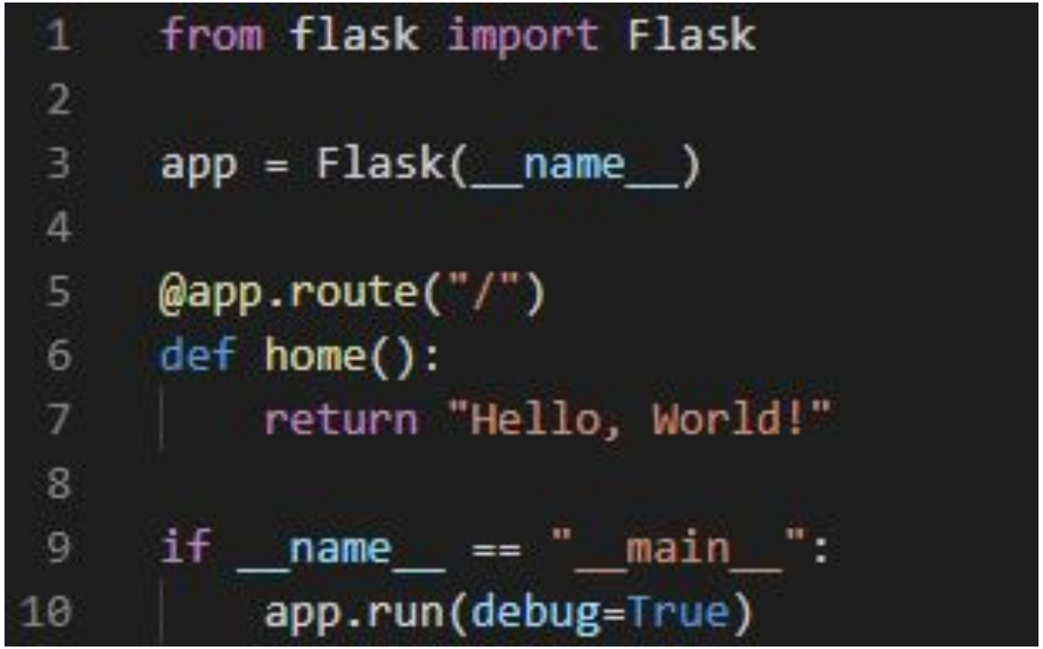
Flask tutorial directory layout



```
FLASK TUTORIALS
  > venv
  main.py
```

How Does a Flask App Work?

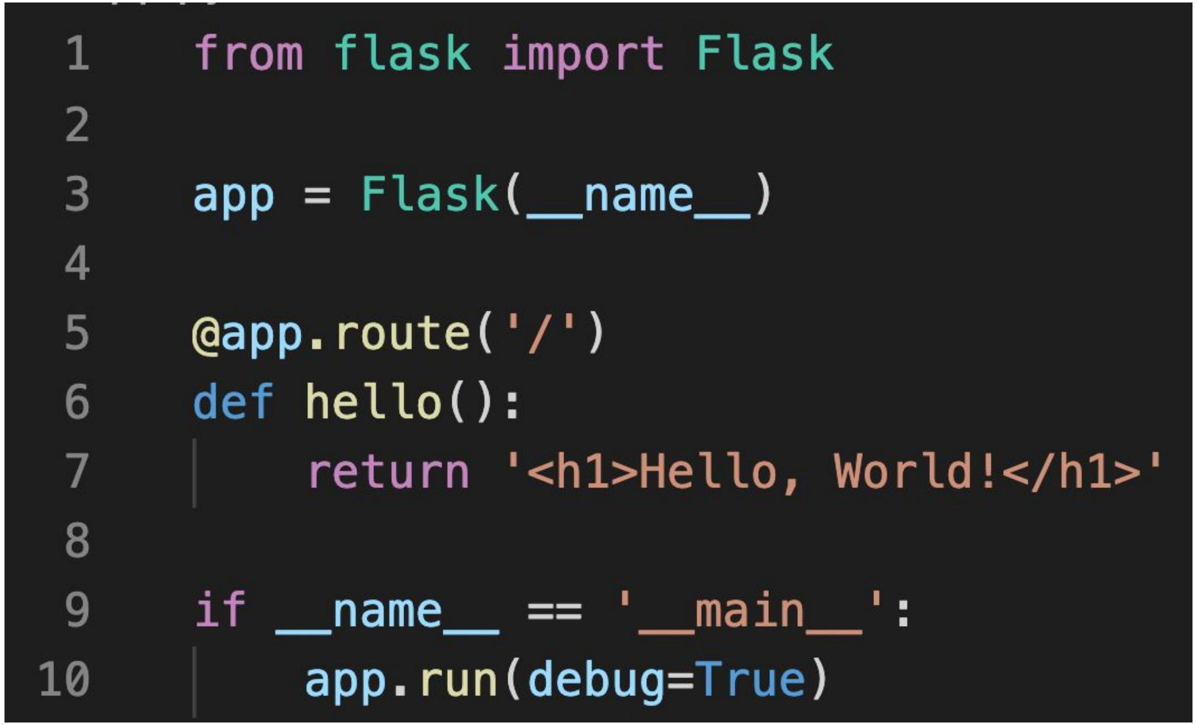
- The code lets us run a basic web application that we can serve, as if it were a website.
- main.py



```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "Hello, World!"

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

How Does a Flask App Work? With HTML code

- Same code with “Hello World” wrapped around in h1 tag
- main.py

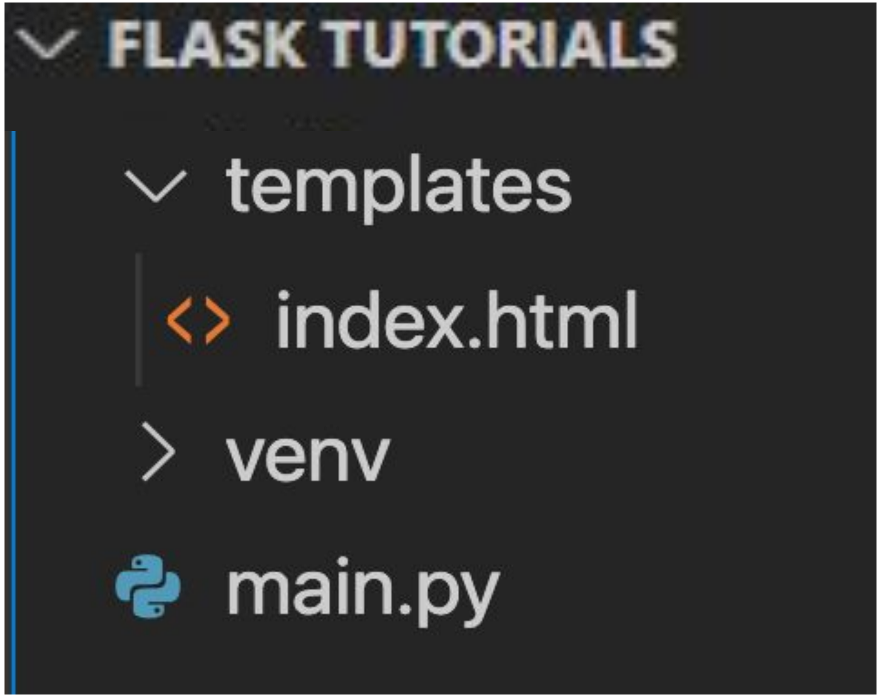


```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return '<h1>Hello, World!</h1>'

if __name__ == '__main__':
    app.run(debug=True)
```

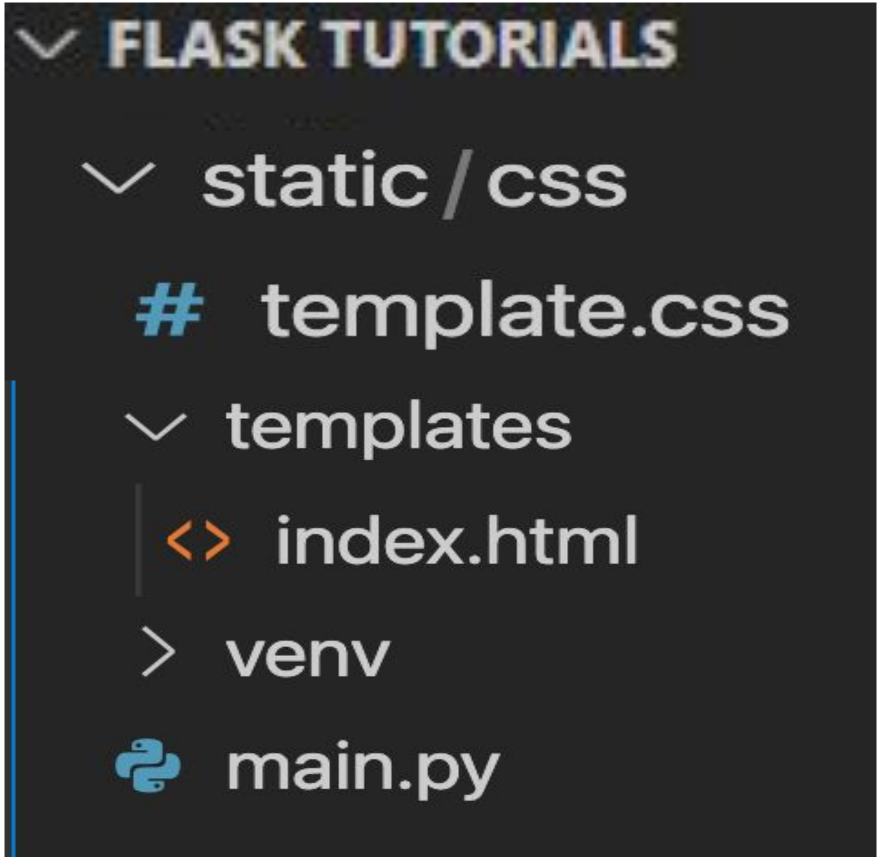
Jinja2 Template Engine- for longer HTML code..

- Jinja2 template is a file that contains the text of a response.
- Text of response is stored in some html file
- Jinja2 template engine lets us render that file
- How?
    - Two-steps
        - Store that html file in templates folder - index.html
        - Use function render_template() provided by Flask to render that file.

Flask tutorial directory layout - with HTML


FLASK TUTORIALS
templates
index.html
venv
main.py

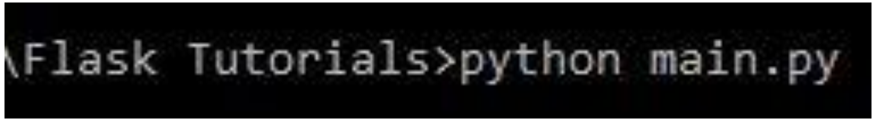
Flask tutorial directory layout - with HTML, CSS



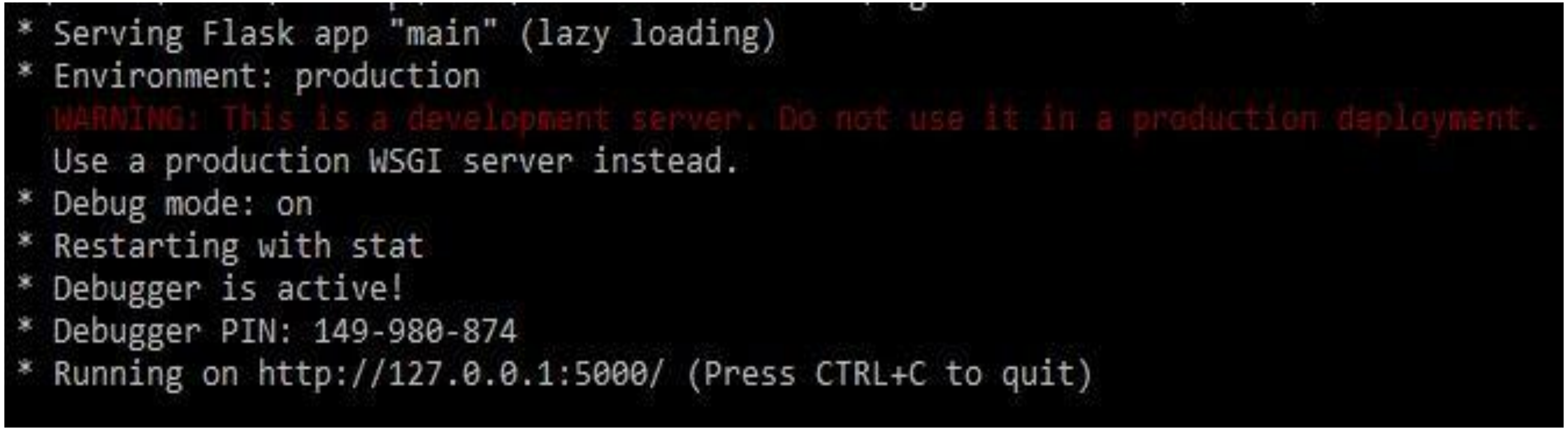
```
FLASK TUTORIALS
  static / css
    template.css
  templates
    index.html
  venv
  main.py
```

Let’s try running main.py

- The code lets us run a basic web application that we can serve, as if it were a website.
- In your Terminal or Command Prompt go to the folder that contains your main.py.
- Then do py main.py or python main.py.



```
\Flask Tutorials>python main.py
```



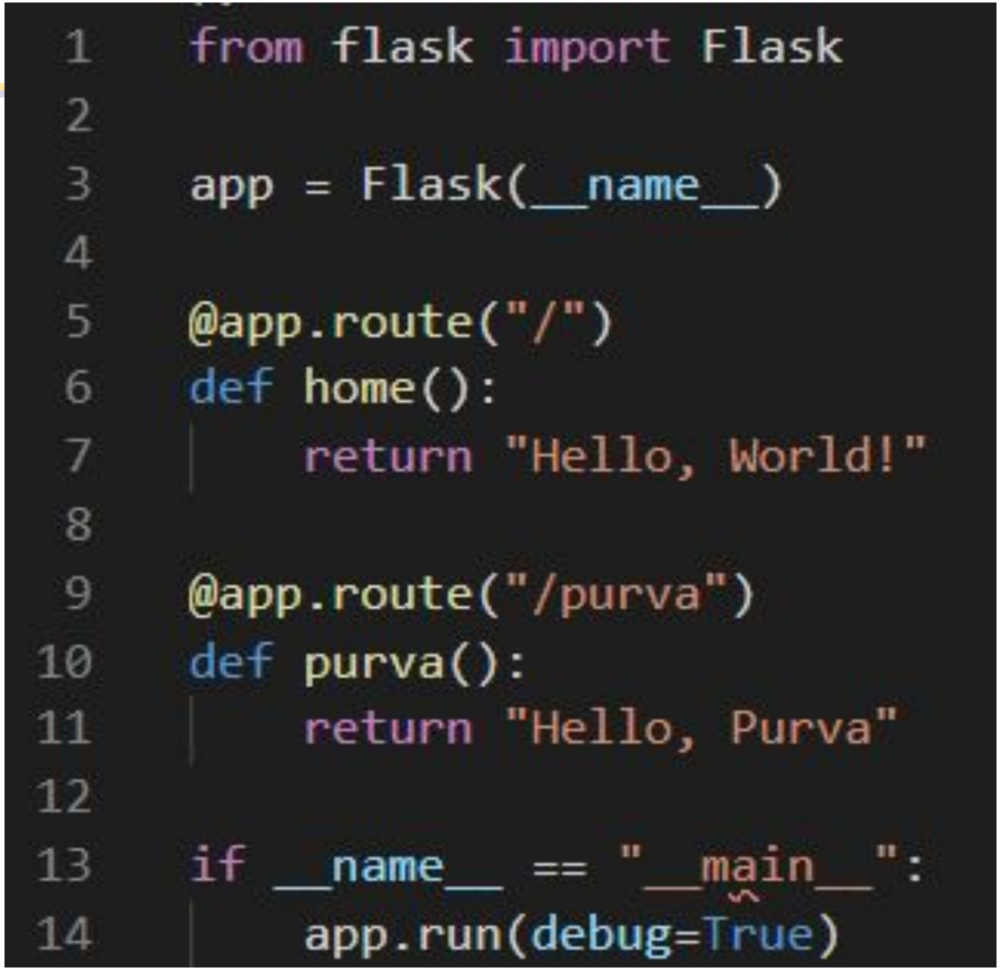
```
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 149-980-874
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

What is localhost?

- The localhost is the default name describing the local computer address also known as the loopback address
- In computer networking talk, localhost refers to “this computer” or even more accurately “the computer I’m working on.”
- Why is localhost useful?
    - Calling your own cellphone to check your set ringtone
- Not much different from you typing in Disney.com or Amazon.com in your browser’s address bar. Every website has its own IP address, but you substitute a “domain name” instead.
- Pretend to be connecting to a Web server or another host computer, but keeping it in-house and close to home by using localhost.

More fun with Flask!!

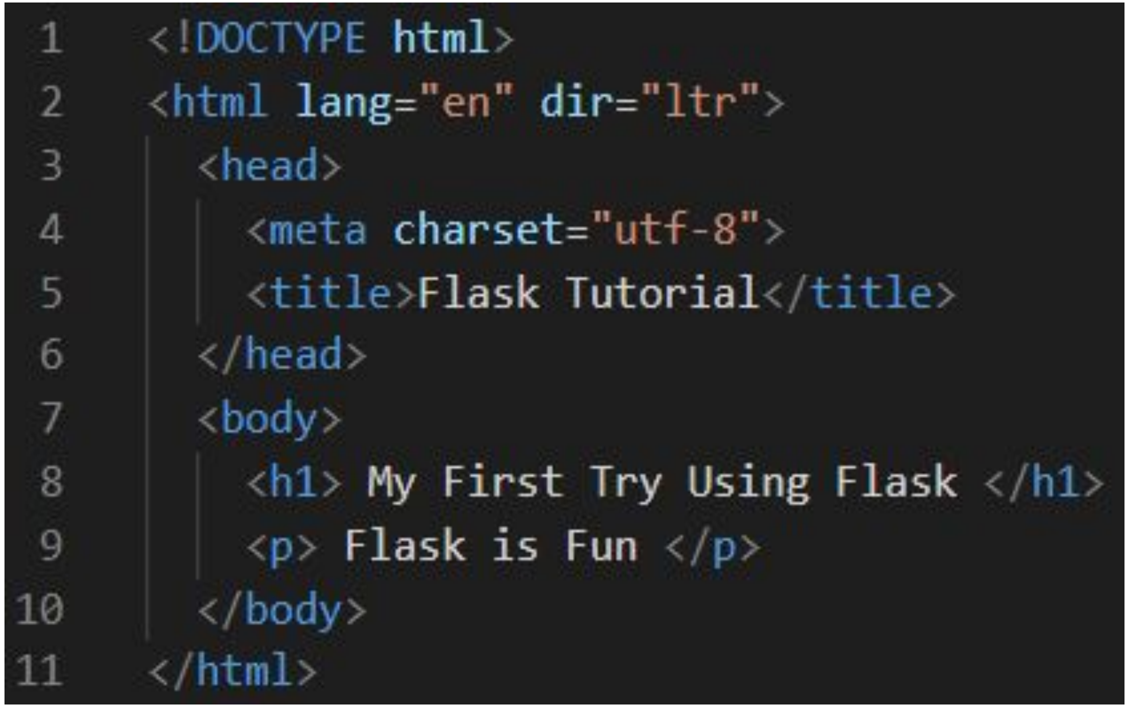
- app.route("/").
- more routes
- main.py



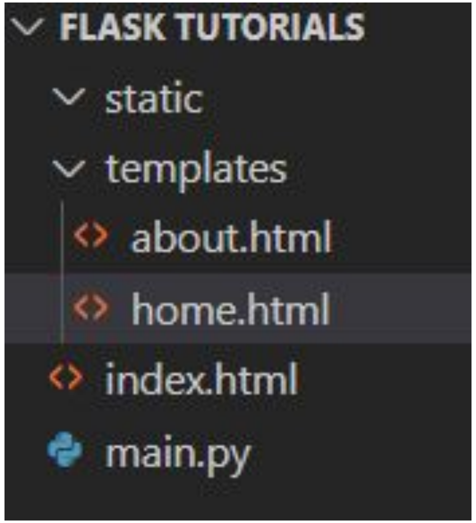
```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "Hello, World!"

@app.route("/purva")
def purva():
    return "Hello, Purva"

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

HTML and CSS

- The code lets us run a basic web application that we can serve, as if it were a website.
- home.html inside templates folder



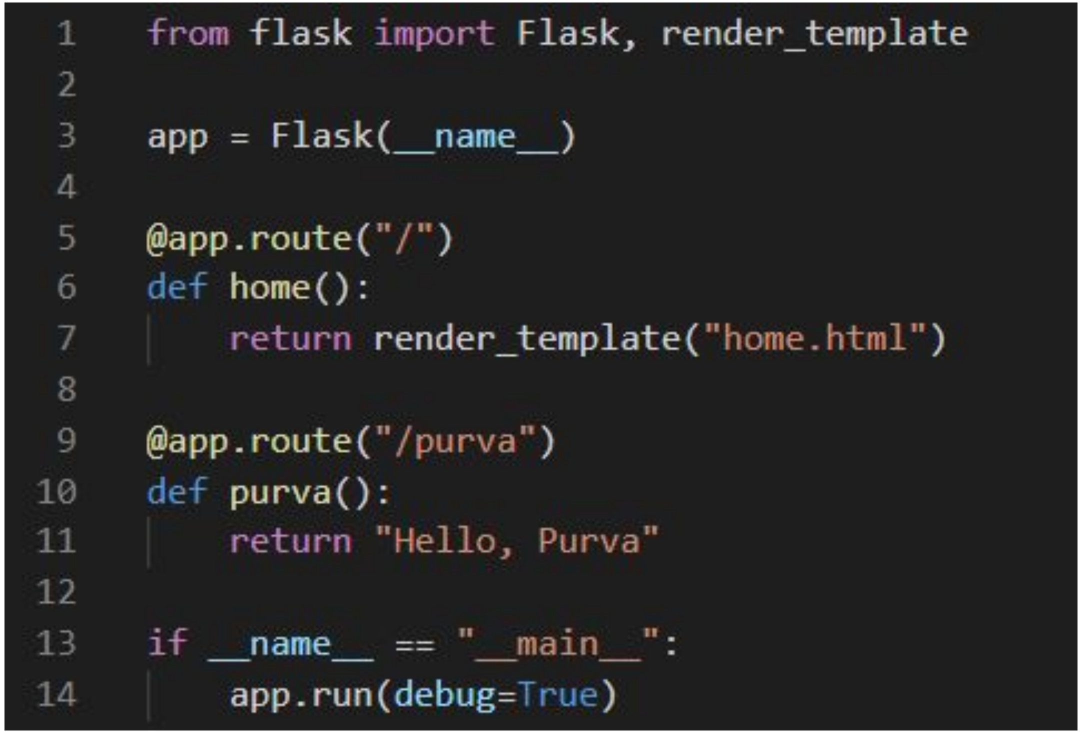
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Flask Tutorial</title>
  </head>
  <body>
    <h1> My First Try Using Flask </h1>
    <p> Flask is Fun </p>
  </body>
</html>
```



```
FLASK TUTORIALS
  static
  templates
    about.html
    home.html
  index.html
  main.py
```

Change main.py

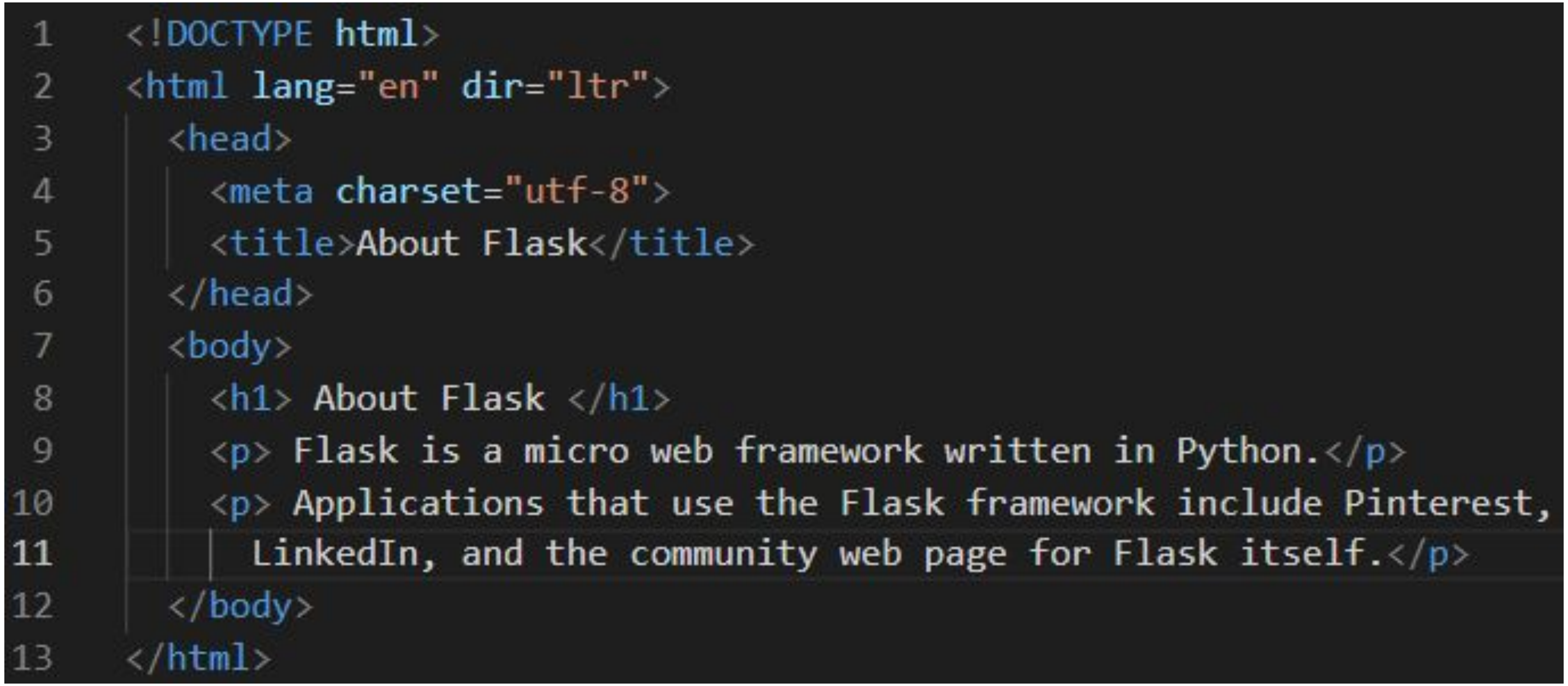
- Changes in main.py
- More info: https://flask.palletsprojects.com/en/1.1.x/quickstart/



```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/purva")
def purva():
    return "Hello, Purva"

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

Let’s Add more html pages : about.html

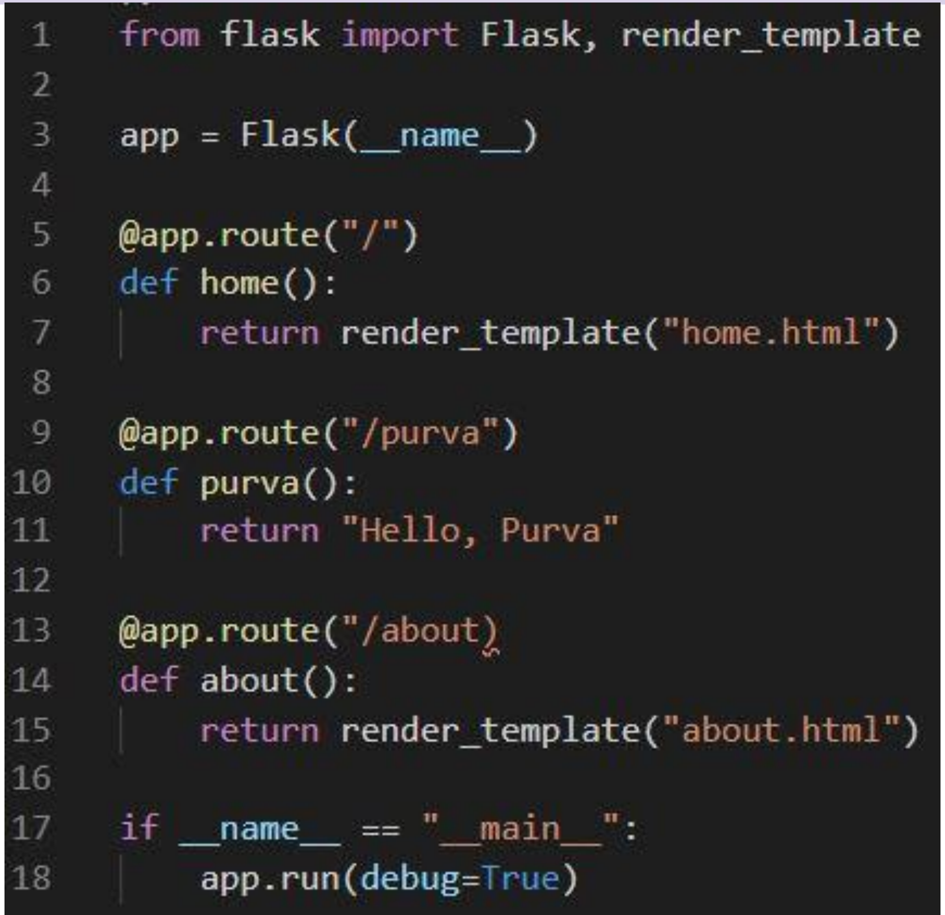
- about.html in templates folder



```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>About Flask</title>
  </head>
  <body>
    <h1> About Flask </h1>
    <p> Flask is a micro web framework written in Python.</p>
    <p> Applications that use the Flask framework include Pinterest,
      LinkedIn, and the community web page for Flask itself.</p>
  </body>
</html>
```

Need to change main.py

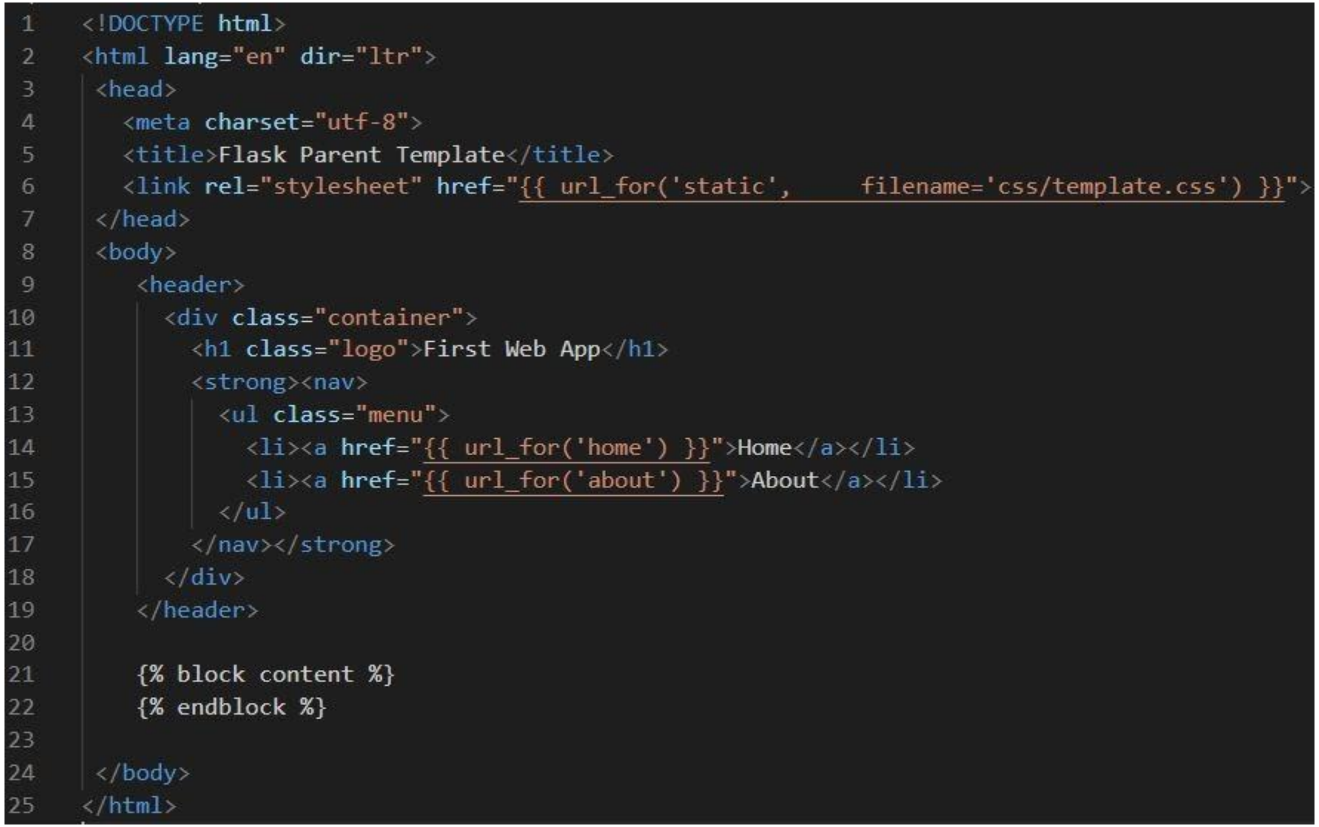
- Changes in main.py



```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/purva")
def purva():
    return "Hello, Purva"

@app.route("/about)
def about():
    return render_template("about.html")

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

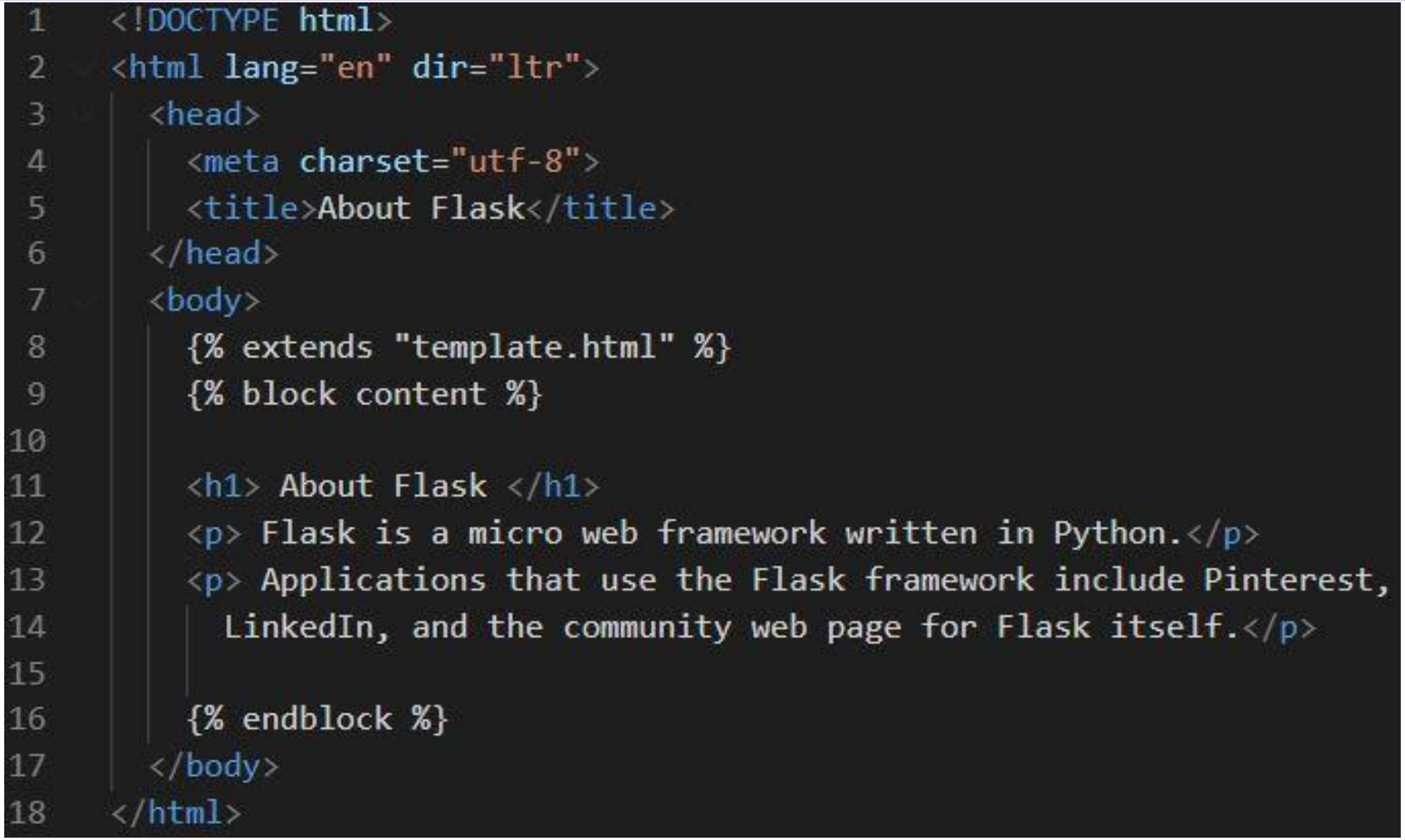
Let’s Connect Both Pages with a Navigation

- template.html: serve as a parent template. Our two child templates will inherit code from it.



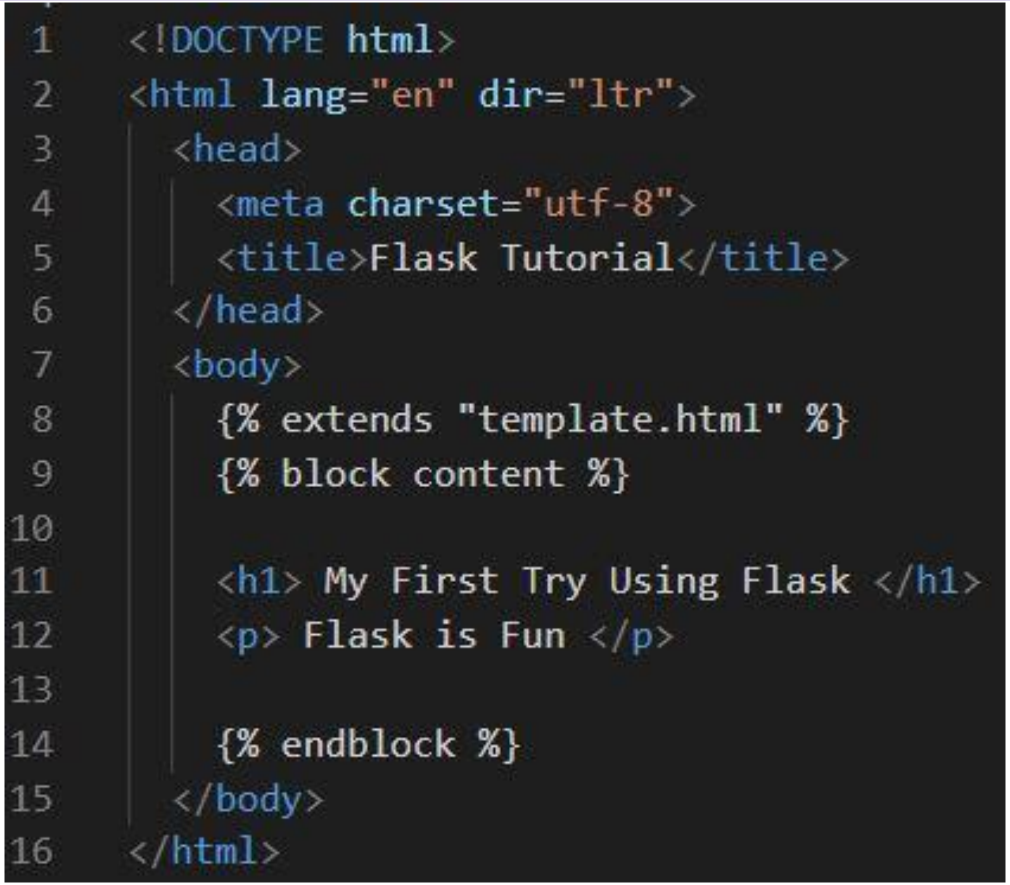
```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
   <meta charset="utf-8">
   <title>Flask Parent Template</title>
   <link rel="stylesheet" href="{{ url_for('static',     filename='css/template.css') }}">
 </head>
 <body>
    <header>
      <div class="container">
        <h1 class="logo">First Web App</h1>
        <strong><nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav></strong>
      </div>
    </header>

    {% block content %}
    {% endblock %}

 </body>
</html>
```

Change about.html



```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>About Flask</title>
  </head>
  <body>
    {% extends "template.html" %}
    {% block content %}

    <h1> About Flask </h1>
    <p> Flask is a micro web framework written in Python.</p>
    <p> Applications that use the Flask framework include Pinterest,
      LinkedIn, and the community web page for Flask itself.</p>

    {% endblock %}
  </body>
</html>
```

Change home.html

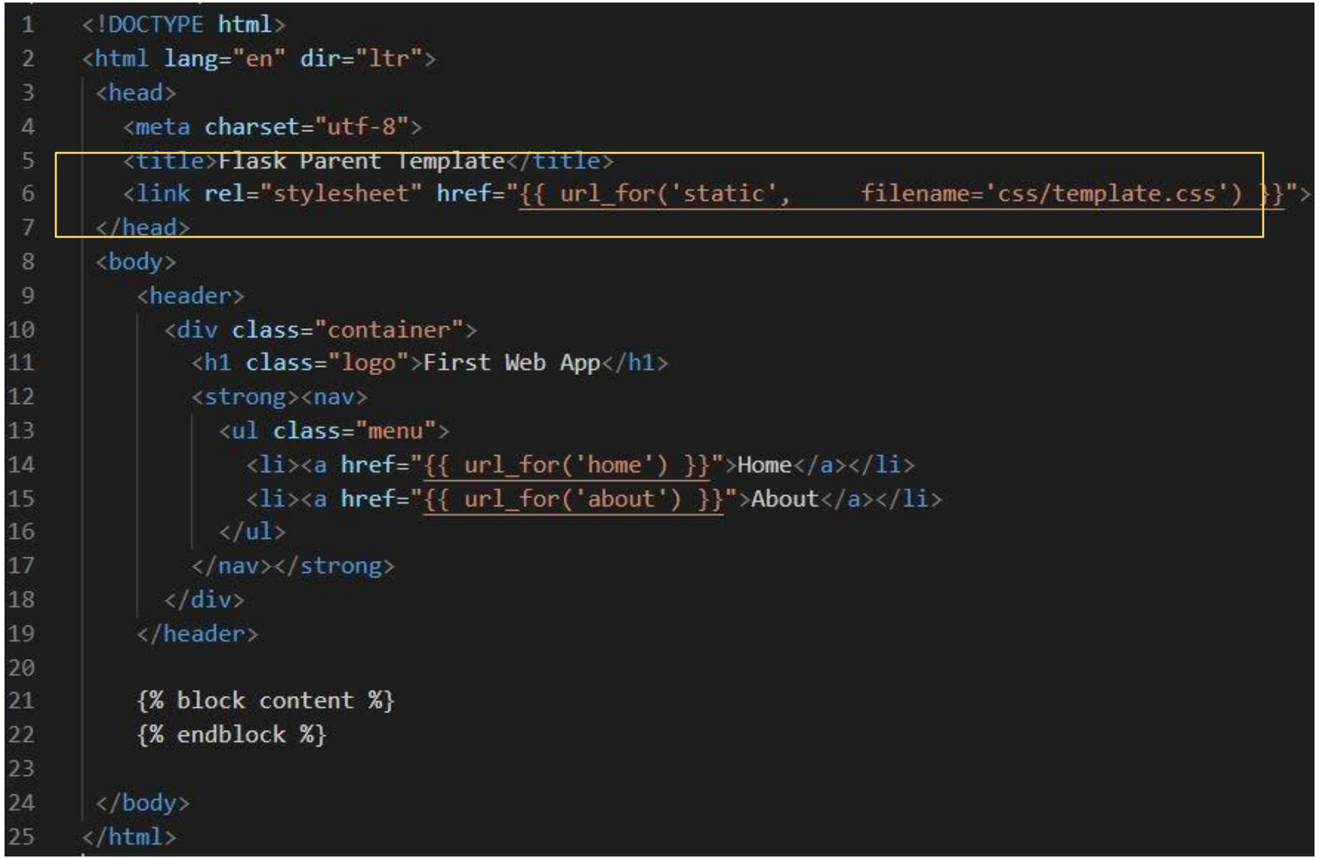


```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Flask Tutorial</title>
  </head>
  <body>
    {% extends "template.html" %}
    {% block content %}

    <h1> My First Try Using Flask </h1>
    <p> Flask is Fun </p>

    {% endblock %}
  </body>
</html>
```

Adding CSS to our website

- Create a folder static
- store CSS, JavaScript, images, and other necessary files
- Linking our CSS with our HTML file
- Our template.html is the one that links all pages.
- We can insert the code here and it will be applicable to all child pages.

CSS added!!



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Flask Parent Template</title>
    <link rel="stylesheet" href="{{ url_for('static',     filename='css/template.css') }}">
  </head>
  <body>
      <header>
        <div class="container">
          <h1 class="logo">First Web App</h1>
          <strong><nav>
            <ul class="menu">
              <li><a href="{{ url_for('home') }}">Home</a></li>
              <li><a href="{{ url_for('about') }}">About</a></li>
            </ul>
          </nav></strong>
        </div>
      </header>

      {% block content %}
      {% endblock %}

  </body>
</html>
```