

BMI CALCULATOR APPLICATION

1. What Are We Building?

We are building a BMI Calculator Web Application.

It is a health-related application where users can:

-  Enter their weight in kilograms
-  Enter their height in centimeters
-  Select their gender
-  Enter their age
-  Calculate their Body Mass Index (BMI)
-  View their BMI category
-  Check their healthy weight range
-  See additional health indicators such as BMI Prime and Ponderal Index

The application runs inside the browser with Flask handling the backend calculations.

2. Technologies Used

Technology	What it is used for
Python	Main programming language
Flask	Creates the backend and handles calculations
HTML	Creates the webpage structure
CSS	Makes the application visually attractive
Jinja2	Displays calculated results dynamically

3. Before Starting (Setup)

Step 3.1: Install Python

Download from:

<https://www.python.org>

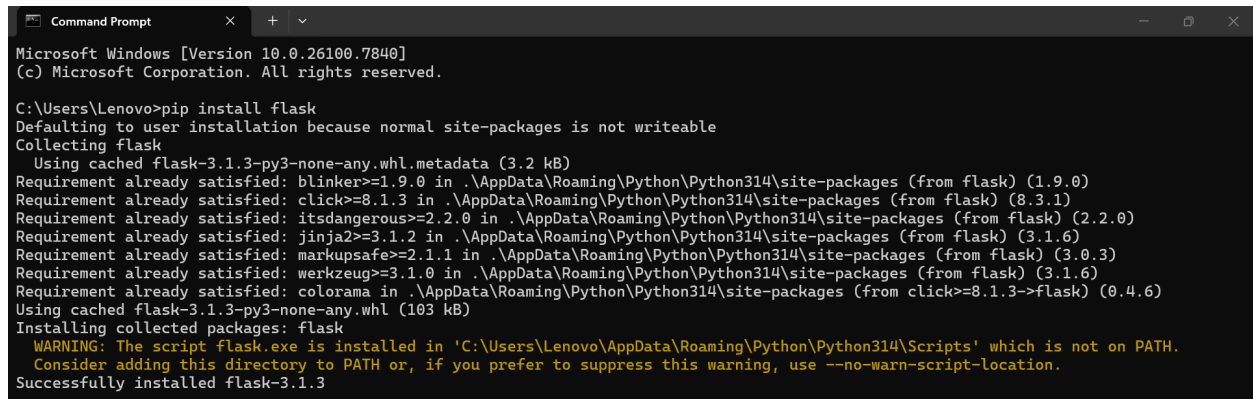
While installing, check:

☒ Add Python to PATH

Step 3.2: Install Flask

Open Command Prompt or Terminal and type:

```
pip install flask
```



```
Microsoft Windows [Version 10.0.26100.7840]
(c) Microsoft Corporation. All rights reserved.

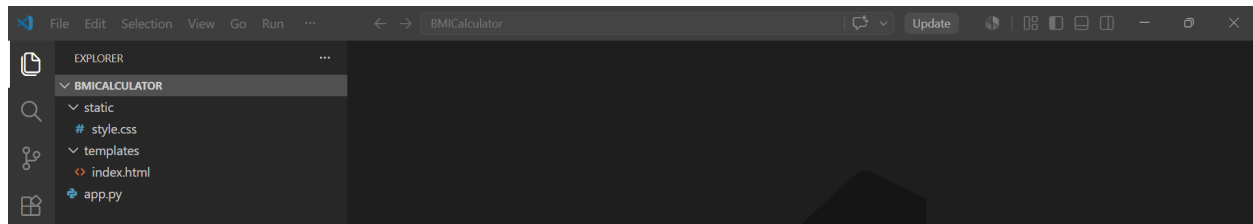
C:\Users\Lenovo>pip install flask
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Using cached flask-3.1.3-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: blinker>=1.9.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (2.2.0)
Requirement already satisfied: Jinja2>=3.1.2 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.1.6)
Requirement already satisfied: MarkupSafe>=2.1.1 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.0.3)
Requirement already satisfied: Werkzeug>=3.1.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.1.6)
Requirement already satisfied: colorama in .\AppData\Roaming\Python\Python314\site-packages (from click>=8.1.3->flask) (0.4.6)
Using cached flask-3.1.3-py3-none-any.whl (103 kB)
Installing collected packages: flask
  WARNING: The script flask.exe is installed in 'C:\Users\Lenovo\AppData\Roaming\Python\Python314\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed flask-3.1.3
```

This installs Flask, which is required to run the application.

4. Create Project Structure

Create a folder named BMICalculator

Inside it, create:



5. Backend Development

File: `app.py`

This file acts as the backend of the application.

It performs:

- User input processing
- BMI calculations
- BMI category determination
- Healthy weight calculations
- Result generation

Step 5.1: Import Required Modules

```
from flask import Flask, render_template, request
```

These help to:

- Create the Flask application
- Display HTML pages
- Receive form data submitted by users

Step 5.2: Create Flask Application

```
app = Flask(__name__)
```

Initializes the Flask application.

Step 5.3: Create Home Route

```
@app.route("/", methods=["GET", "POST"])  
def index():
```

This route:

- Displays the calculator page

- Accepts user input
- Performs BMI calculations

Step 5.4: Receive User Inputs

```
age = int(request.form["age"])
gender = request.form["gender"]
height_cm = float(request.form["height"])
weight = float(request.form["weight"])
```

Collects:

- Age
- Gender
- Height
- Weight

entered by the user.

Step 5.5: Convert Height

```
height_m = height_cm / 100
```

Converts height from centimeters to meters.

Example:

170 cm → 1.70 m

Step 5.6: Calculate BMI

```
bmi = weight / (height_m ** 2)
```

BMI Formula:

$\text{BMI} = \text{Weight (kg)} \div \text{Height}^2 \text{ (m}^2\text{)}$

Step 5.7: Determine BMI Category

```
if bmi < 18.5:
```

```

        category = "Underweight"
elif bmi < 25:
    category = "Normal"
elif bmi < 30:
    category = "Overweight"
else:
    category = "Obese"

```

Categories:

BMI Range	Category
Below 18.5	Underweight
18.5 – 24.9	Normal
25 – 29.9	Overweight
30 and above	Obese

Step 5.8: Calculate Healthy Weight Range

```

min_weight = 18.5 * (height_m ** 2)
max_weight = 25 * (height_m ** 2)

```

This calculates the healthy weight range for the user's height.

Step 5.9: Calculate BMI Prime

```

bmi_prime = bmi / 25

```

BMI Prime compares the user's BMI with the upper limit of the healthy BMI range.

Interpretation:

- Less than 0.74 → Underweight
- 0.74 – 1.00 → Healthy
- Above 1.00 → Overweight

Step 5.10: Calculate Ponderal Index

```
ponderal = weight / (height_m ** 3)
```

Ponderal Index is another measurement used to assess body composition.

Step 5.11: Store Results

```
result = {  
    "bmi": round(bmi,1),  
    "category": category,  
    "min_weight": round(min_weight,1),  
    "max_weight": round(max_weight,1),  
    "bmi_prime": round(bmi_prime,2),  
    "ponderal": round(ponderal,2)  
}
```

Stores all calculated values.

Step 5.12: Display Results

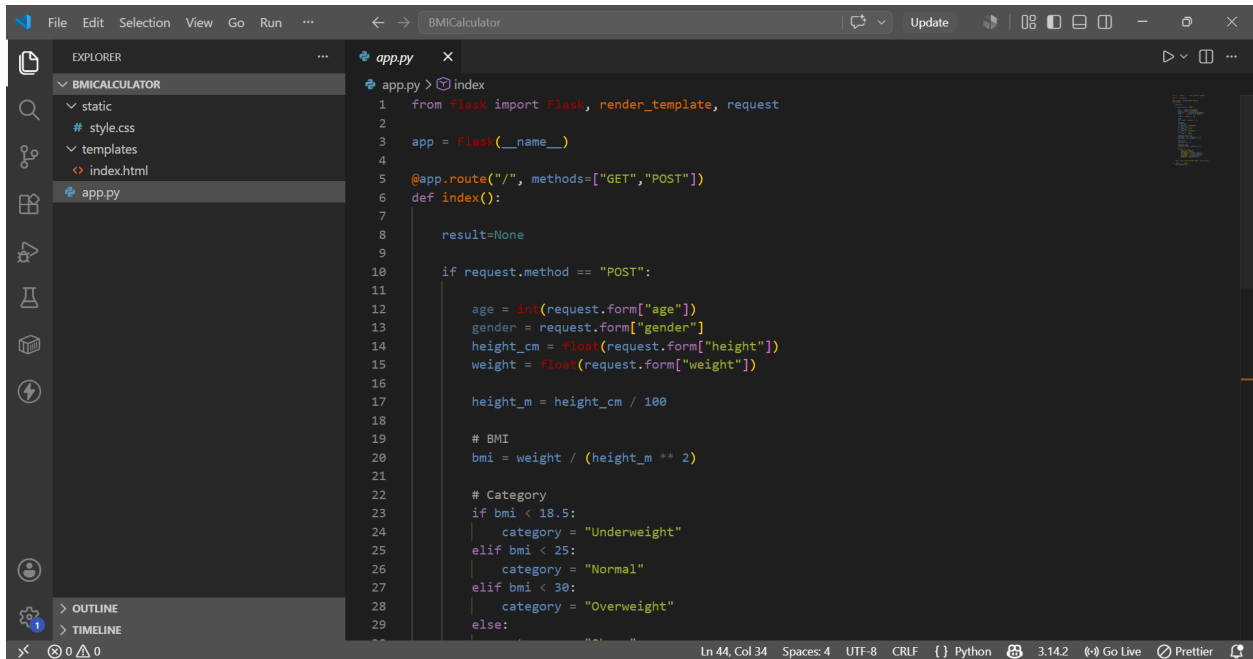
```
return render_template("index.html", result=result)
```

Sends the results to the webpage.

Step 5.13: Run Application

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

Starts the Flask server.



6. Frontend Development

File: index.html

This file creates:

- Input form
- Gender selection
- Calculate button
- Results section

Step 6.1: Add Page Title

```
<title>BMI Calculator</title>
```

Set the browser tab title.

Step 6.2: Connect CSS File

```
<link  
rel="stylesheet"  
href="{{ url_for('static', filename='style.css') }}"
```

```
/>
```

Links the stylesheet.

Step 6.3: Create Main Container

```
<div class="container">
```

Holds all page elements.

Step 6.4: Add Heading

```
<h1>BMI Calculator</h1>
```

Displays the application title.

Step 6.5: Create Form

```
<form method="POST">
```

Allows users to submit their details.

Step 6.6: Age Input

```
<input type="number" name="age" required />
```

Collects user's age.

Step 6.7: Gender Selection

```
<input type="radio" name="gender" value="male" required />
```

```
<input type="radio" name="gender" value="female" />
```

Allows users to select gender.

Step 6.8: Height Input

```
<input  
type="number"  
step="any"  
name="height"  
placeholder="Height (cm) "  
required  
>
```

Collects height in centimeters.

Step 6.9: Weight Input

```
<input  
type="number"  
step="any"  
name="weight"  
placeholder="Weight (kg) "  
required  
>
```

Collects weight in kilograms.

Step 6.10: Calculate Button

```
<button type="submit">  
Calculate  
</button>
```

Submit the form.

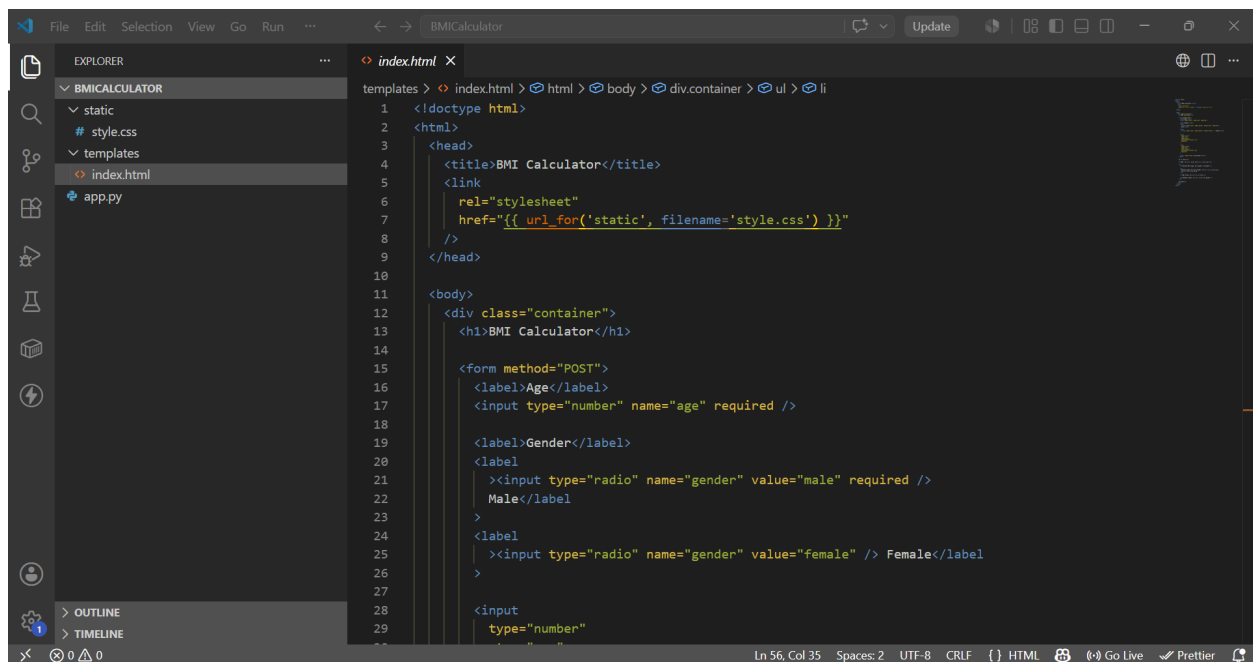
Step 6.11: Display Results

```
{% if result %}
```

Shows results only after calculation.

Displays:

- BMI value
- BMI category
- Healthy BMI range
- Healthy weight range
- BMI Prime
- Ponderal Index



```
1 <!doctype html>
2 <html>
3 <head>
4 <title>BMI Calculator</title>
5 <link
6   rel="stylesheet"
7   href="{{ url_for('static', filename='style.css') }}"
8 />
9 </head>
10
11 <body>
12 <div class="container">
13 <h1>BMI Calculator</h1>
14
15 <form method="POST">
16 <label>Age</label>
17 <input type="number" name="age" required />
18
19 <label>Gender</label>
20 <label
21   ><input type="radio" name="gender" value="male" required />
22   Male</label>
23 >
24 <label
25   ><input type="radio" name="gender" value="female" /> Female</label>
26 >
27
28 <input
29   type="number"
```

7. Styling

File: style.css

This file improves the appearance of the calculator.

Step 7.1: Style Page

```
body {
    font-family: Arial;
```

```
}
```

Uses Arial font throughout the application.

Step 7.2: Add Background Gradient

```
background:
linear-gradient(
120deg,
#f6d365,
#fda085
);
```

Creates an attractive background.

Step 7.3: Center Content

```
display: flex;
justify-content: center;
align-items: center;
```

Center the calculator on the page.

Step 7.4: Style Container

```
.container {
  background: white;
}
```

Creates a card-like appearance.

Additional styling includes:

- Rounded corners
- Padding
- Shadow effects

Step 7.5: Style Input Fields

```
input {  
  padding: 10px;  
}
```

Improves usability.

Additional styling adds:

- Rounded borders
- Proper spacing
- Consistent sizing

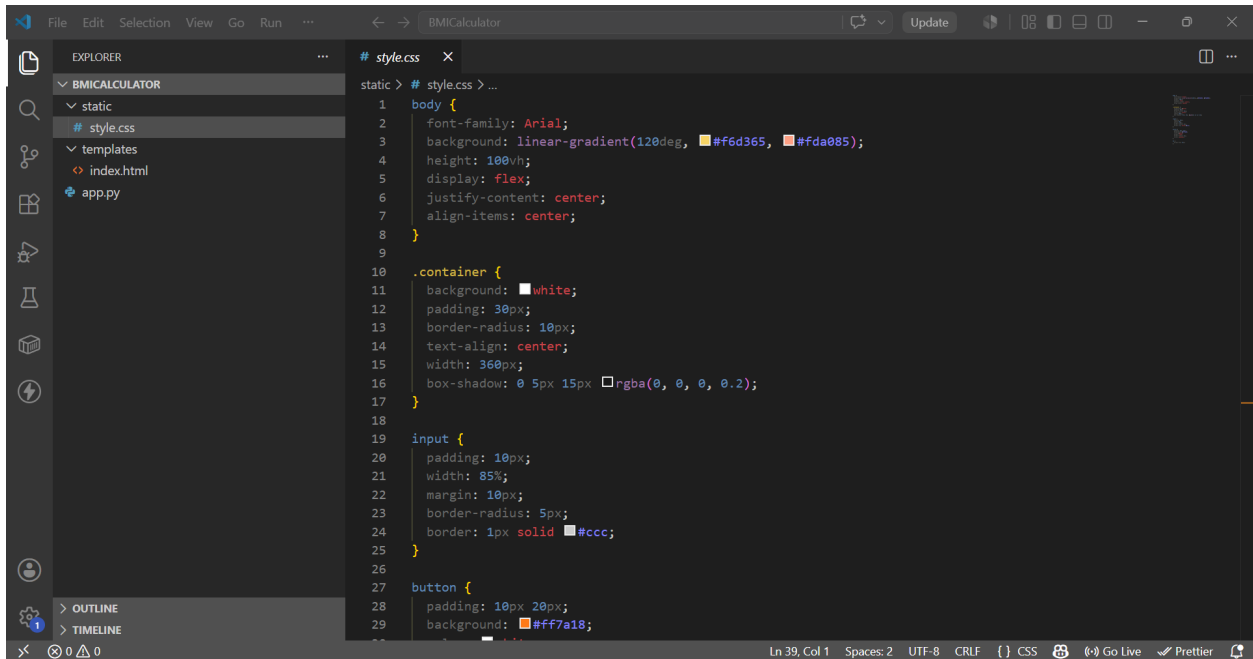
Step 7.6: Style Button

```
button {  
  background: #ff7a18;  
  color: white;  
}
```

Creates an attractive Calculate button.

Features include:

- Rounded corners
- Pointer cursor
- Better visual appeal



The screenshot shows the Visual Studio Code editor with the 'BMICalculator' project open. The Explorer sidebar on the left shows the file structure: 'static' folder containing 'style.css', 'templates' folder containing 'index.html', and 'app.py'. The main editor area displays the content of 'style.css'.

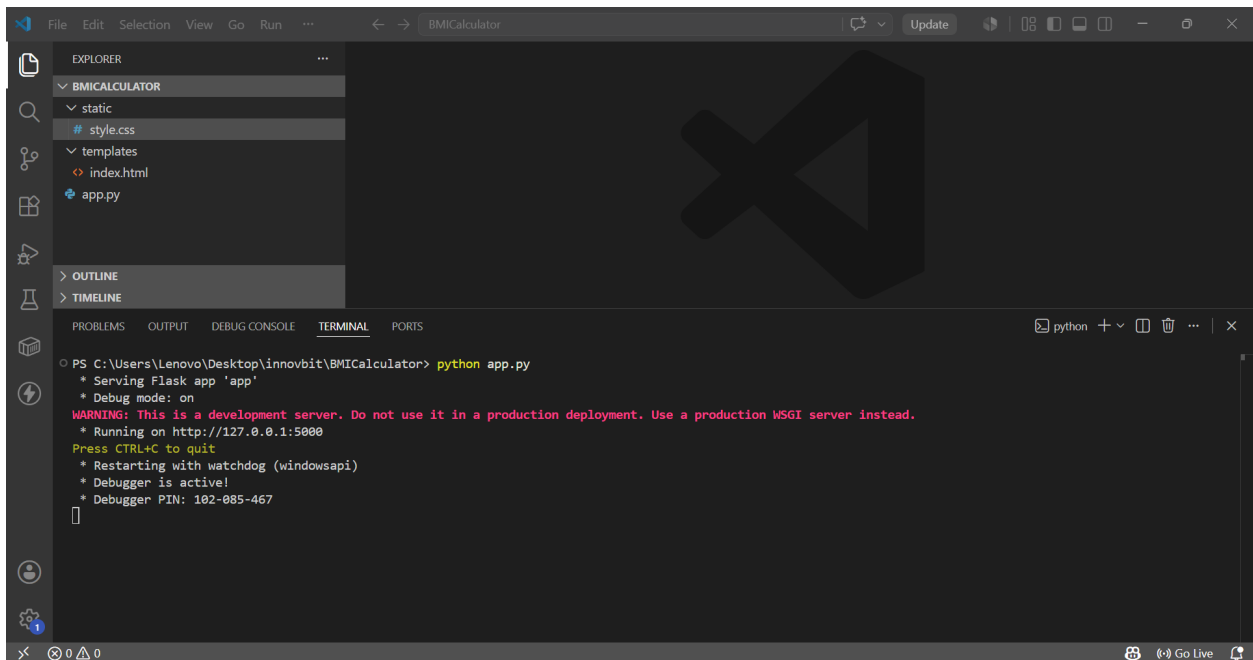
```
# style.css
static > # style.css > ...
1 body {
2   font-family: Arial;
3   background: linear-gradient(120deg, #fed365, #fda085);
4   height: 100vh;
5   display: flex;
6   justify-content: center;
7   align-items: center;
8 }
9
10 .container {
11   background: white;
12   padding: 30px;
13   border-radius: 10px;
14   text-align: center;
15   width: 360px;
16   box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
17 }
18
19 input {
20   padding: 10px;
21   width: 85%;
22   margin: 10px;
23   border-radius: 5px;
24   border: 1px solid #ccc;
25 }
26
27 button {
28   padding: 10px 20px;
29   background: #ff7a18;
```

The status bar at the bottom indicates 'Ln 39, Col 1', 'Spaces: 2', 'UTF-8', 'CRLF', and 'CSS'.

8. Run the App

Open terminal inside the project folder:

```
python app.py
```



The screenshot shows the Visual Studio Code editor with the 'BMICalculator' project open. The Explorer sidebar on the left shows the file structure: 'static' folder containing 'style.css', 'templates' folder containing 'index.html', and 'app.py'. The main editor area is empty, displaying a large 'X' watermark. The Terminal panel at the bottom shows the output of running 'python app.py'.

```
PS C:\Users\Lenovo\Desktop\innovbit\BMICalculator> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 102-085-467
```

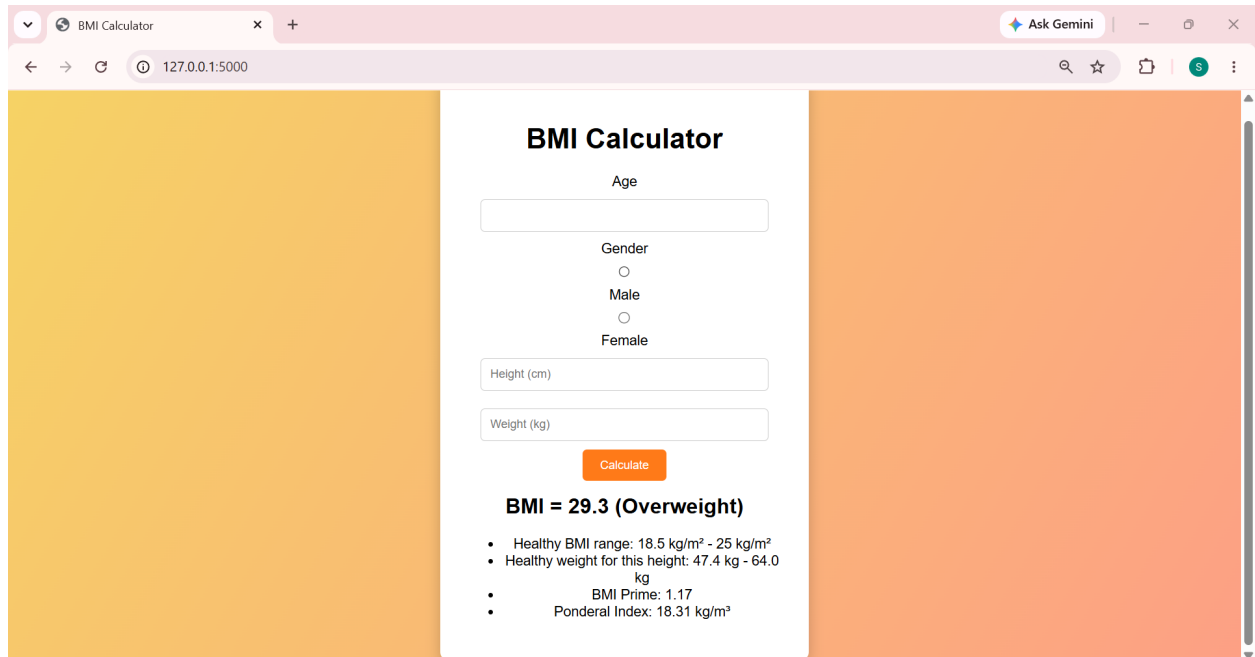
The status bar at the bottom indicates 'python' and 'Go Live'.

Then open the browser and visit:
`http://127.0.0.1:5000`

The screenshot shows a web browser window with the title "BMI Calculator". The address bar displays "127.0.0.1:5000". The page has a light orange background. In the center, there is a white card with the title "BMI Calculator". Below the title, there are input fields for "Age", "Height (cm)", and "Weight (kg)". There are also radio buttons for "Gender" with options "Male" and "Female". An orange "Calculate" button is at the bottom of the card.

The screenshot shows the same web browser window, but now the input fields are filled with values: "Age" is 40, "Height (cm)" is 160, and "Weight (kg)" is 75. The "Male" radio button is selected. The orange "Calculate" button is still present. Below the input fields, the result is displayed: "BMI = 29.3 (Overweight)". Below this, there is a list of bullet points providing additional information:

- Healthy BMI range: 18.5 kg/m² - 25 kg/m²
- Healthy weight for this height: 47.4 kg - 64.0 kg
- BMI Prime: 1.17
- Ponderal Index: 18.31 kg/m³



BMI Calculator

Age

Gender

☐ Male

☐ Female

Height (cm)

Weight (kg)

Calculate

BMI = 29.3 (Overweight)

- Healthy BMI range: 18.5 kg/m² - 25 kg/m²
- Healthy weight for this height: 47.4 kg - 64.0 kg
- BMI Prime: 1.17
- Ponderal Index: 18.31 kg/m³

9. How Everything Works Together (Flow)

- The user opens the website.
- Flask loads `index.html`.
- User enters:
 - Age
 - Gender
 - Height
 - Weight
- User clicks Calculate.
- Form data is sent to Flask.
- Flask calculates:
 - BMI
 - BMI category
 - Healthy weight range
 - BMI Prime
 - Ponderal Index
- Results are stored.

- Results are displayed on the webpage.

10. Common Mistakes

- Forgetting to install Flask
- Incorrect folder names (templates, static)
- Not connecting the CSS file properly
- Using wrong input names
- Forgetting to convert height from cm to meters
- Not running app.py from the project directory
- Forgetting debug=True during development

11. Tips & Tricks

11.1 Add Input Validation

Prevent negative values for:

- Height
- Weight
- Age

11.2 Add Reset Button

Example:

```
<button type="reset">
```

Clear

```
</button>
```

Clears all inputs.

11.3 Improve UI

You can:

- Use Bootstrap
- Add icons

- Add animations
- Include charts

11.4 Add Health Recommendations

Provide suggestions based on BMI category.

Example:

- Underweight → Nutritional guidance
- Overweight → Exercise recommendations

11.5 Store User History

Save previous BMI records using:

- SQLite
- MySQL
- Local Storage

12. Final Summary

This project teaches:

- Flask basics
- HTML forms
- CSS styling
- Handling POST requests
- Processing user input
- Performing health-related calculations
- Dynamic webpage rendering using Jinja2

It is an excellent beginner project for learning how frontend and backend technologies work together to build practical healthcare applications.