

EVEN ODD CHECKER APPLICATION

1. What Are We Building?

We are building an Even Odd Checker Web Application.

It is a number-based application where users can:

-  Enter any integer value
-  Check whether the number is Even or Odd
-  Receive instant results
-  Access the application directly through a web browser
-  Learn the concept of number divisibility in an interactive way

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

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>

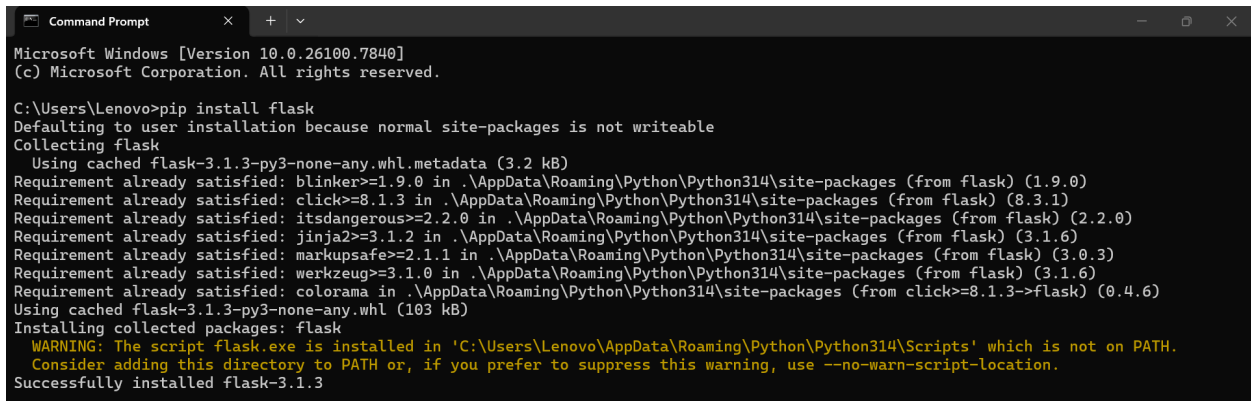
While installing, check:

✓ 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.

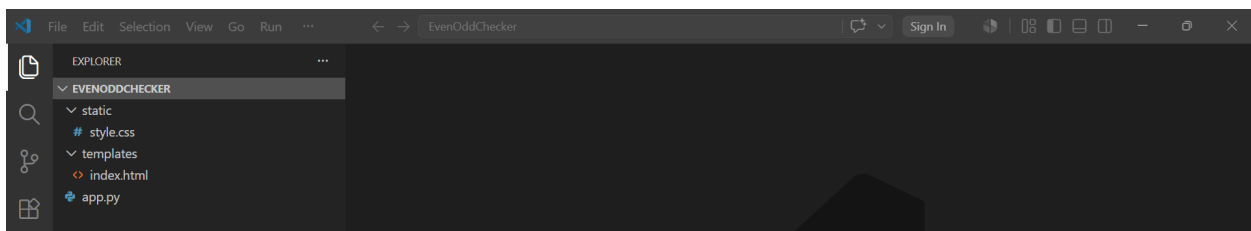
C:\Users\Lenovo>pip install flask
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Using cached flask-3.1.3-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: blinker>=1.9.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (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 EvenOddChecker

Inside it, create:



5. Backend Development

File: app.py

This file acts as the backend of the application.

It performs:

- User input processing
- Even/Odd calculations
- Result generation
- Rendering webpages

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 checker page
- Accepts user input
- Performs Even/Odd checking

Step 5.4: Receive User Input

```
number = int(request.form["number"])
```

Collects the number entered by the user.

Example:

- 12
- 45
- 101

Step 5.5: Check Whether Number is Even or Odd

```
if number % 2 == 0:
    result = "Even"
else:
    result = "Odd"
```

Logic Used:

A number is Even if it is perfectly divisible by 2.

A number is Odd if it leaves a remainder of 1 when divided by 2.

Step 5.6: Examples

Number	Calculation	Result
10	$10 \% 2 = 0$	Even
15	$15 \% 2 = 1$	Odd
28	$28 \% 2 = 0$	Even
99	$99 \% 2 = 1$	Odd

Step 5.7: Store Results

```
result = {
    "number": number,
    "type": result
}
```

Stores:

- The entered number
- The calculated result

Step 5.8: Display Results

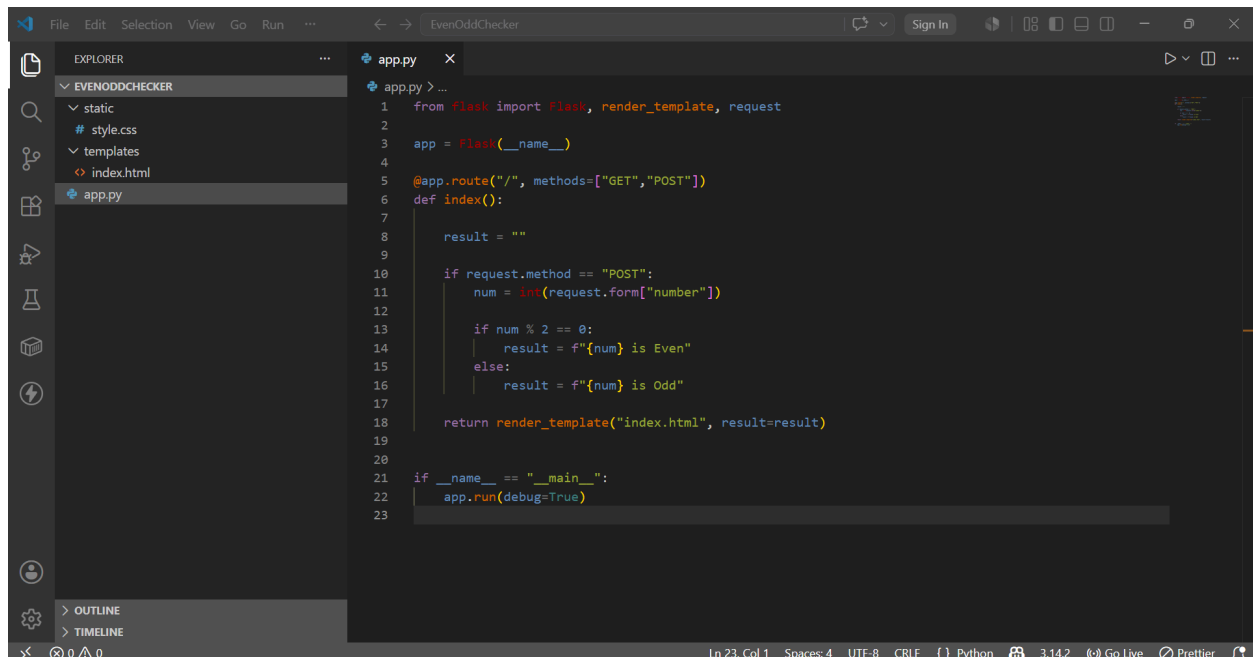
```
return render_template(  
    "index.html",  
    result=result  
)
```

Send the results to the webpage.

Step 5.9: Run Application

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

Starts the Flask server.



6. Frontend Development

File: `index.html`

This file creates:

- Input form
- Check button
- Results section

Step 6.1: Add Page Title

```
<title>Even Odd Checker</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>Even Odd Checker</h1>
```

Displays the application title.

Step 6.5: Create Form

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

Allows users to submit a number.

Step 6.6: Number Input

```
<input  
type="number"  
name="number"  
placeholder="Enter a number"  
required  
>
```

Collects the user's number.

Examples:

- 7
- 22
- 105

Step 6.7: Check Button

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

Submits the form for processing.

Step 6.8: Display Results

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

Shows results only after checking.

Displays:

- The entered number
- Whether it is Even or Odd

Example:

Number: 18

Result: Even Number

```

1 <!doctype html>
2 <html>
3   <head>
4     <title>Even Odd Checker</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>Even Odd Checker</h1>
14
15       <form method="POST">
16         <input
17           type="number"
18           name="number"
19           placeholder="Enter number"
20           required
21         />
22
23         <button type="submit">Check</button>
24       </form>
25
26       <h2>{{ result }}</h2>
27     </div>
28   </body>
29 </html>

```

7. Styling

File: style.css

This file improves the appearance of the checker.

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,
#84fab0,
#8fd3f4
);
```

Creates an attractive background.

Step 7.3: Center Content

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

Centers the checker 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: #4CAF50;  
    color: white;  
}
```

Creates an attractive Check button.

Features include:

- Rounded corners
- Pointer cursor
- Better visual appeal

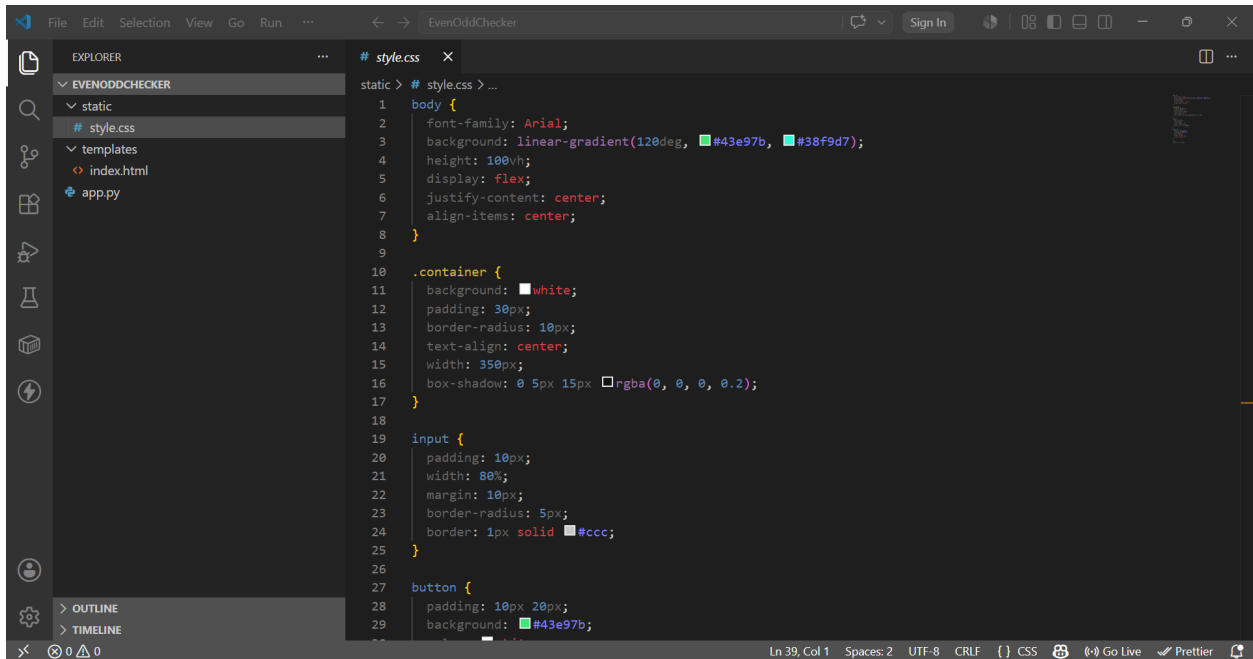
Step 7.7: Style Result Section

```
.result {  
    font-weight: bold;  
}
```

Highlights the final result.

Additional styling may include:

- Larger text size
- Different colors for Even and Odd results
- Margin spacing



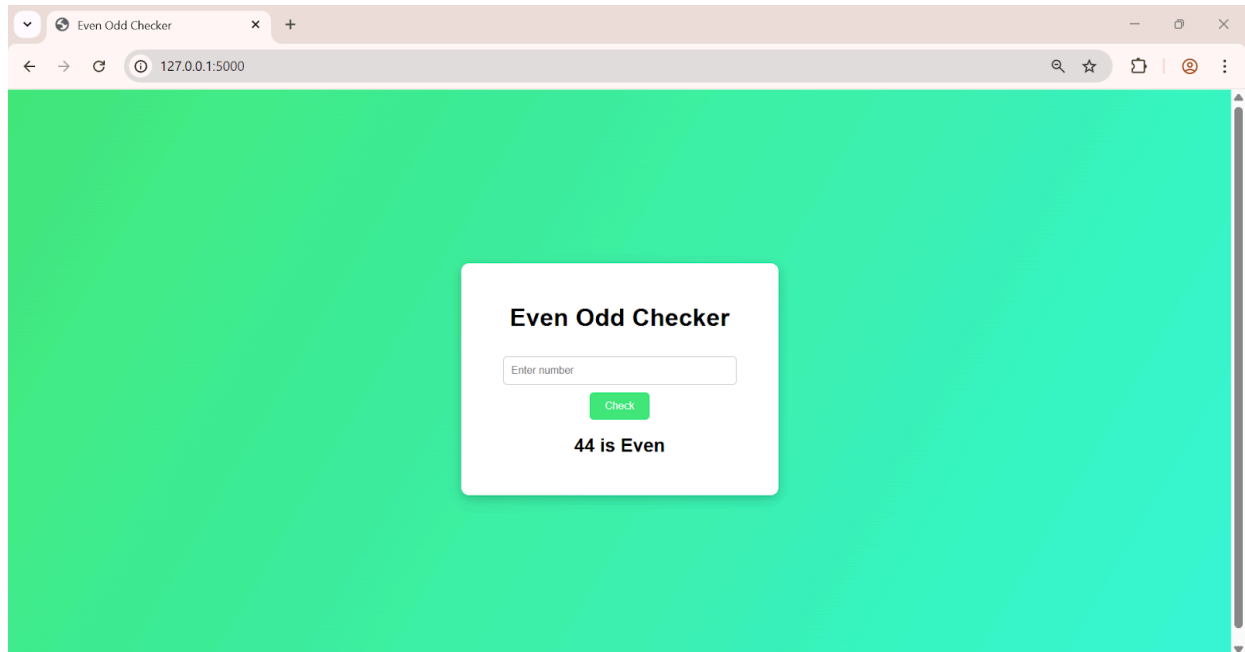
8. Run the App

Open terminal inside the project folder:

```
python app.py
```

Then open the browser and visit:

```
http://127.0.0.1:5000
```



9. How Everything Works Together (Flow)

- The user opens the website.
- Flask loads index.html.
- The user enters a number.
- User clicks Check.
- Form data is sent to Flask.
- Flask receives the number.
- Flask checks whether the number is divisible by 2.
- The result (Even or Odd) is generated.
- The result is stored.
- The result is displayed on the webpage.

10. Common Mistakes

- Forgetting to install Flask
- Incorrect folder names (templates, static)
- Not connecting the CSS file properly
- Using incorrect input names

- Forgetting to convert input into an integer
- Running app.py outside the project directory
- Forgetting debug=True during development

11. Tips & Tricks

11.1 Add Input Validation

Prevent empty submissions and invalid entries.

11.2 Add Reset Button

Example:

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

Clears the entered number.

11.3 Improve UI

You can:

- Use Bootstrap
- Add icons
- Add animations
- Include colorful result cards

11.4 Display Mathematical Explanation

Example:

$17 \div 2$ leaves a remainder of 1.

Therefore, 17 is an Odd number.

Helps beginners understand the concept.

11.5 Maintain Check History

Save previously checked numbers using:

- SQLite
- MySQL
- Local Storage

12. Final Summary

This project teaches:

- Flask basics
- HTML forms
- CSS styling
- Handling POST requests
- Processing user input
- Conditional statements in Python
- Dynamic webpage rendering using Jinja2

It is an excellent beginner project for learning how frontend and backend technologies work together to build practical educational applications. The Even Odd Checker provides a simple and interactive way to understand one of the fundamental concepts of number systems using web technologies.