2. Explain what an XSS attack is and how it can be prevented in a web application. Write a program in Python using Flask that demonstrates how to prevent XSS attacks.

PREREQUISITES:

- Python 3.x installed on the machine
- Flask library installed
- A web browser to access the web application

SETUP:

1. Install Python on the machine if it is not already installed.
2. Install the Flask library using pip command: `pip install flask`
3. Copy the code into a Python file with a .py extension.
4. Run the Python file using the command `python <filename>.py`
5. Open a web browser and go to `http://localhost:5000` to access the web application.

ANSWER:

An XSS (Cross-Site Scripting) attack is a type of web security vulnerability where an attacker injects malicious scripts into a web page viewed by other users. This can allow the attacker to steal sensitive information, hijack user accounts, or perform other malicious actions.

To prevent XSS attacks, web developers should follow these best practices:

1. Input Validation: Validate and sanitize all input, including user-generated content, to prevent attackers from injecting malicious scripts.
2. Output Encoding: Use output encoding to convert special characters to their corresponding HTML entities, such as converting "<" to "<" and ">" to ">".
3. Content Security Policy (CSP): Implement a Content Security Policy (CSP) to restrict the sources of executable scripts in a web page.
4. HttpOnly and Secure Flags: Use the HttpOnly and Secure flags when setting cookies to prevent them from being accessed by malicious scripts.
5. Frameworks and Libraries: Use frameworks and libraries that have built-in protections against XSS attacks, such as Django, Ruby on Rails, or React.

By implementing these measures, web developers can significantly reduce the risk of XSS attacks in their web applications

CODE:

```
from flask import Flask, request

app = Flask(__name__)

@app. route ("/")
def index ():
    name = request.args.get ("name", "")
    return f"""
        <html>
            <head>
                <title>XSS Example</title>
            </head>
            <body>
                <h1>Hello, {escape_html(name)}! </h1>
                <form action="/" method="GET">
                    <label for="name">Enter your name:</label>
                    <input type="text" id="name" name="name"
value="{escape_html(name)}">
                    <input type="submit" value="Submit">
                </form>
            </body>
        </html>
    """

def escape_html(s):

    return s. replace ("&", "&amp;"). replace ("<", "&lt;"). replace (">", "&gt;").
replace ("", "&quot;"). replace ("'", "&#39;")

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

EXPLANATION:

This is a simple Flask web application that demonstrates how to prevent cross-site scripting (XSS) attacks by properly escaping user input in HTML templates.

The application defines a single route ("/") that displays a simple HTML form asking the user to enter their name. When the user submits the form, their name is included in the query string of the request. The `name` parameter is then retrieved from the request using `request.args.get("name", "")`.

Before rendering the response, the `escape_html` function is called to escape any potentially malicious characters in the user's name. The function replaces all occurrences of "&", "<", ">", double quotes, and single quotes with their corresponding HTML entities. This helps prevent XSS attacks by ensuring that any user-supplied input is displayed as plain text in the web page, rather than being interpreted as HTML or JavaScript code.

Finally, the `index` function returns an HTML response containing the escaped user name, along with a form for entering a new name. The `value` attribute of the input field is also set to the escaped user name, so that the user's previously-entered name is pre-populated in the form.

If the user does not supply a name parameter in the query string, an empty string is used as the default value for the `name` variable.

The `if __name__ == "__main__":` statement at the bottom of the script ensures that the application is only run when the script is executed directly, and not when it is imported as a module. When the script is executed, the Flask application is started by calling `app.run()`

PROGRAM OUTCOMES:

- The program creates a simple Flask web application that takes user input and displays it on a webpage.
- The `index()` function takes the value of the `name` parameter from the URL using the `request.args.get()` method, and displays it on the webpage.
- The `escape_html()` function is used to prevent XSS attacks by replacing certain characters with their corresponding HTML entities.
- When the user enters their name in the input field and submits the form, the web application displays a personalized greeting message with the user's name.