Flask Basics

1. Setting Up a Basic Flask Application

Installation: First, you need to install Flask. You can do this using `pip`:
bash

```
pip install Flask
```

-

Creating the Application: Start by creating a new file, e.g., `app.py`. This file will contain your Flask application.

```
from flask import Flask

app = Flask(__name__)

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

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

-

Running the Application: You can run your Flask application by executing the file:
bash

```
 app.py
```

- By default, Flask will run the application on `http://127.0.0.1:5000/`.
- **Understanding the Code**:
  - `from flask import Flask`: Import the Flask class from the `flask` module.
  - `app = Flask(__name__)`: Create a Flask application instance.
  - `@app.route('/')`: Define a route that maps to the home page (`/`).

    - `def home():`: The function that handles requests to the specified route.
    - `app.run(debug=True)`: Start the Flask development server, with `debug=True` to enable debug mode.

2. Understanding Routes

- **What are Routes?**: Routes in Flask are URL patterns that are associated with specific functions. When a user visits a URL, Flask will call the function associated with that route.

Defining Routes: You can define multiple routes in your Flask application. Each route corresponds to a URL pattern.

```
@app.route('/')
def home():
    return "Welcome to the Home Page"

@app.route('/about')
def about():
    return "This is the About Page"
```

-

Dynamic Routes: Routes can also be dynamic, allowing you to capture parts of the URL and pass them to your view function.

```
@app.route('/user/<username>')
def show_user_profile(username):
    return f"User: {username}"
```

-
    - Here, `'<username>'` is a dynamic part of the URL, and the value is passed to the `show_user_profile` function.

Route Methods: By default, routes handle `GET` requests, but you can specify other HTTP methods like `POST`, `PUT`, `DELETE`, etc.

```
@app.route('/submit', methods=['POST'])
```

```
def submit():
    return "Form Submitted"
```

-

3. Handling HTTP Requests

- **Handling Different HTTP Methods**: Flask allows you to handle different HTTP methods (like `GET`, `POST`, `PUT`, `DELETE`) within your route functions.

GET Request: This is the default method, typically used to request data.

```
@app.route('/greet', methods=['GET'])
def greet():
    return "Hello, World!"
```

  -

POST Request: Typically used to submit data to the server.

```
from flask import request

@app.route('/submit', methods=['POST'])
def submit():
    data = request.form['data']
    return f"Data received: {data}"
```

  -
- **Accessing Request Data**:

Query Parameters: Accessed using `request.args`.

```
@app.route('/search')
def search():
    query = request.args.get('q')
    return f"Search query: {query}"
```

  -

Form Data: Accessed using `request.form`.

```
@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    return f"Logged in as: {username}"
```

  -

JSON Data: Accessed using `request.json`.

```
@app.route('/api/data', methods=['POST'])
def api_data():
    json_data = request.json
    return f"Received JSON: {json_data}"
```

  -
- **Returning Responses**:
  - You can return strings, JSON, or even HTML templates as responses.

```
@app.route('/json')
def json_response():
    return {"message": "This is a JSON response"}
```

-

Summary

- **Setting Up**: Install Flask, create an application instance, define routes, and run the development server.
- **Routes**: URL patterns that map to functions in your Flask app; can be static or dynamic.
- **HTTP Requests**: Handle various HTTP methods (`GET`, `POST`, etc.) and access data from requests using `request.args`, `request.form`, and `request.json`.