**Q-1) Explain the concept of state in React and how it is used to manage component data.**

* State is a built-in object in React that allows a component to hold and manage dynamic data.
* Unlike props, state is managed within the component itself.
* When state changes, React automatically re-renders the component to reflect the updated data.

**Key Characteristics of State:**

* Mutable (can change over time).
* Local to the component (not directly accessible by other components).
* Triggers UI updates when changed.

**Using State in Functional Components:**

* With hooks (useState):

import React, { useState } from "react";

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

* Here, count is state, and setCount is used to update it.

**Using State in Class Components:**

class Counter extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

Render () {

return (

<div>

<p>Count: {this.state.count}</p>

<button onClick={() => this.setState ({ count: this.state.count + 1} )} >

Increment

</button>

</div>

);

}

}

* this.state stores the component’s data.
* Changing this.state triggers re-render.

**Q-2) Why is this.setState() used in class components, and how does it work?**

**Purpose of this.setState():**

* In class components, you cannot directly modify this.state.
* Example (wrong):

this.state.count = this.state.count + 1;

* Instead, React provides this.setState() to update state safely and trigger re-rendering.

**How this.setState() works:**

* Takes an object or a function as an argument.
* Merges the new state with the existing state.
* Automatically re-renders the component with updated state.

**Example (Object form):**

this.setState({ count: this.state.count + 1 });

**Example (Function form - safer for async updates):**

this.setState((prevState) => ({

count: prevState.count + 1

}));

**Key Points:**

* Always use this.setState() to update state in class components.
* It ensures state consistency and proper component re-rendering.
* Directly changing this.state won’t update the UI.