1. Explain the difference between Stateless and Stateful widgets with examples.

Answer:

- **StatelessWidget**
  - **A widget that does not change its state once built.**
  - **It is immutable – the UI remains the same until the widget is rebuilt from outside.**
  - **Example:**
  - **class MyStateless extends StatelessWidget {**
  - **@override**
  - **Widget build(BuildContext context) {**
  - **return Center(**
  - **child: Text("I am Stateless"),**
  - **);**
  - **}**
  - **}**
- **StatefulWidget**
  - **A widget that can change its state dynamically during runtime.**
  - **It is mutable and maintains its state using a State class.**
  - **Example:**
  - **class MyStateful extends StatefulWidget {**
  - **@override**
  - **State<MyStateful> createState() => _MyStatefulState();**
  - **}**
  -
  - **class _MyStatefulState extends State<MyStateful> {**
  - **int counter = 0;**
  -
  - **@override**

```
Widget build(BuildContext context) {
 return Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
   Text("Counter: $counter"),
   ElevatedButton(
    onPressed: () {
     setState(() {
      counter++;
     });
    },
    child: Text("Increment"),
   )
  ],
 );
 }
}
```

👉 **Difference in short:**

- **Stateless → Fixed UI, no internal state.**
- **Stateful → Dynamic UI, updates with setState().**

---

2. Describe the widget lifecycle and how state is managed in Stateful widgets.

Answer:

- **Widget Lifecycle (StatefulWidget):**
  1. **createState() → Called when the widget is created.**
  2. **initState() → Called once when the widget is inserted into the widget tree. Good for initializing variables or fetching data.**
  3. **build() → Called to render the widget. Runs every time setState() is called.**

4. **didUpdateWidget() → Called when the widget is rebuilt with new data.**
5. **dispose() → Called when the widget is removed from the widget tree. Used for cleanup (like closing streams, controllers).**

- **State Management in StatefulWidget:**
  - **The State object holds the data that can change.**
  - **When data changes, setState() is called.**
  - **setState() triggers the build() method again to reflect changes on screen.**

Example lifecycle snippet:

```
class MyStateful extends StatefulWidget {
 @override
 State<MyStateful> createState() => _MyStatefulState();
}

class _MyStatefulState extends State<MyStateful> {
 @override
 void initState() {
  super.initState();
  print("initState called");
 }

 @override
 Widget build(BuildContext context) {
  return Text("Hello Lifecycle");
 }

 @override
 void dispose() {
  super.dispose();
```

```
  print("dispose called");
 }
}
```

---

3. List and describe five common Flutter layout widgets (e.g., Container, Column, Row).

Answer:

1. **Container**
    - **A versatile box model widget.**
    - **Can add padding, margins, borders, background colors, or constraints.**
    - **Example:**
    - **Container(**
    - **color: Colors.blue,**
    - **padding: EdgeInsets.all(10),**
    - **child: Text("Inside Container"),**
    - **)**
2. **Column**
    - **Arranges widgets vertically.**
    - **Example:**
    - **Column(**
    - **children: [Text("One"), Text("Two")],**
    - **)**
3. **Row**
    - **Arranges widgets horizontally.**
    - **Example:**
    - **Row(**
    - **children: [Icon(Icons.star), Text("Star")],**
    - **)**
4. **Stack**

- **Places widgets on top of each other (overlapping).**
- **Useful for overlays.**
- **Example:**
- **Stack(**
- **children: [**
- **Image.network("https://via.placeholder.com/150"),**
- **Positioned(bottom: 10, right: 10, child: Text("Overlay"))**
- **],**
- **)**

5. **ListView**
    - **Displays a scrollable list of widgets.**
    - **Example:**
    - **ListView(**
    - **children: [Text("Item 1"), Text("Item 2"), Text("Item 3")],**

)