Code Coverage Explained

1. What is Code Coverage?

- **Definition**: Code coverage is a software quality metric that measures how much of the source code is executed when automated tests (like unit tests) are run.
- **Purpose**: Helps identify untested parts of code and ensures testing efforts are effective.
- **Example**: If a program has 100 lines of code, and tests execute 80 of them, the code coverage is **80%**.

2. How is Code Coverage Measured?

Code coverage is usually measured as a **percentage**:

$$\text{Code Coverage (\%)} = \frac{\text{Number of lines (or blocks/branches) executed by tests}}{\text{Total number of lines (or blocks/branches)}} \times 100$$

Different measurement criteria include:

- **Line Coverage** → % of executed lines.
- **Branch Coverage** → % of decision branches (if/else, switch cases) executed.
- **Function/Method Coverage** → % of functions/methods invoked.
- **Statement Coverage** → % of executable statements run.
- **Condition Coverage** → % of boolean expressions evaluated as both true and false.

3. Code Coverage vs Test Coverage

Aspect	Code Coverage	Test Coverage
Definition	How much code is executed during testing.	How much of the software requirements, features, or scenarios are tested.
Focus	Focuses on source code execution.	Focuses on test cases vs requirements.
Measurement	% of code lines/branches	% of requirements/features covered by

	executed.	tests.
Example	"80% of code lines are tested."	"All login scenarios are tested (positive & negative)."
Relation	Technical, developer-centric.	Functional, QA-centric.

Both are important: High code coverage doesn't guarantee that the software requirements are tested thoroughly, and high test coverage doesn't mean the code itself is fully exercised.

---

4. Code Coverage Techniques

Different techniques provide more detailed insight:

1. **Function Coverage** – Checks if each function/method was called.
2. **Statement Coverage** – Checks if each line was executed.
3. **Branch Coverage** – Ensures all decision points (if, switch) are tested.
4. **Condition Coverage** – Validates each boolean sub-expression (x > 0, y == true) is both true and false.
5. **Path Coverage** – Tests all possible execution paths (very exhaustive, often impractical for large codebases).

---

5. Code Coverage Tools

Popular tools by ecosystem:

- **Java** → JaCoCo, Cobertura
- **JavaScript/Node.js** → Istanbul (nyc), Jest (built-in), Mocha + NYC
- **Python** → Coverage.py
- **C/C++** → gcov, lcov, BullseyeCoverage
- **C#/.NET** → Visual Studio Code Coverage, OpenCover
- **General/CI Integration** → SonarQube, Codecov, Coveralls

- Code coverage ensures your code is being executed by tests.
- Test coverage ensures your **requirements and scenarios** are validated.
- Using **both** gives the most complete view of quality.