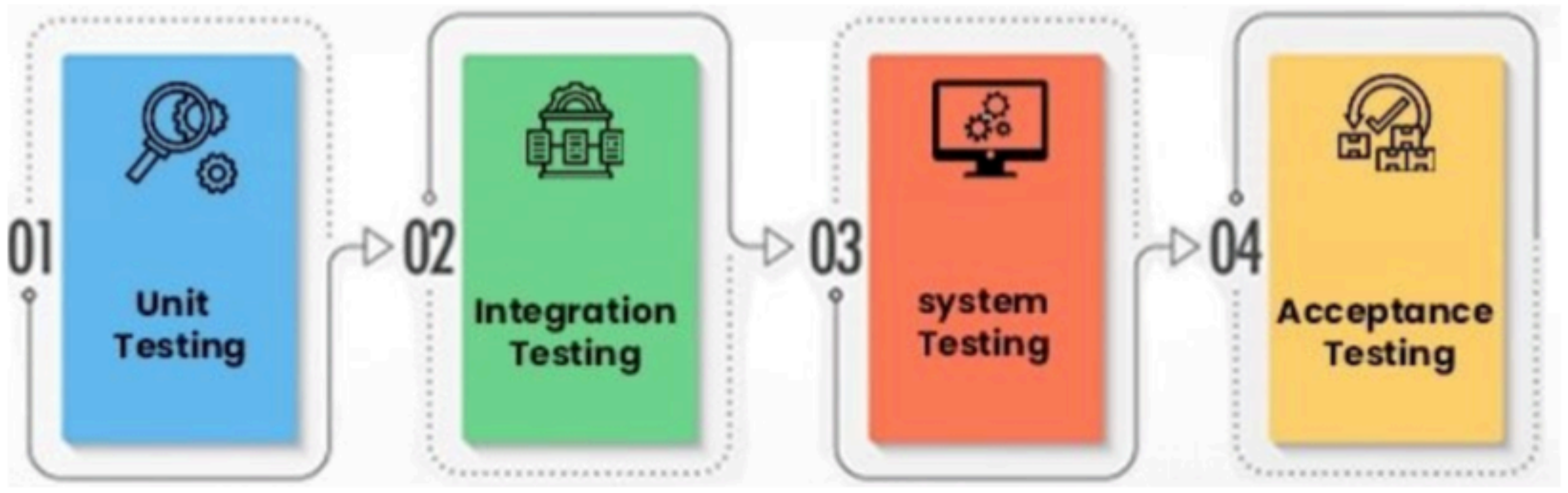
Testing Levels



Interview Question & answers in this article:

What are the four testing levels?

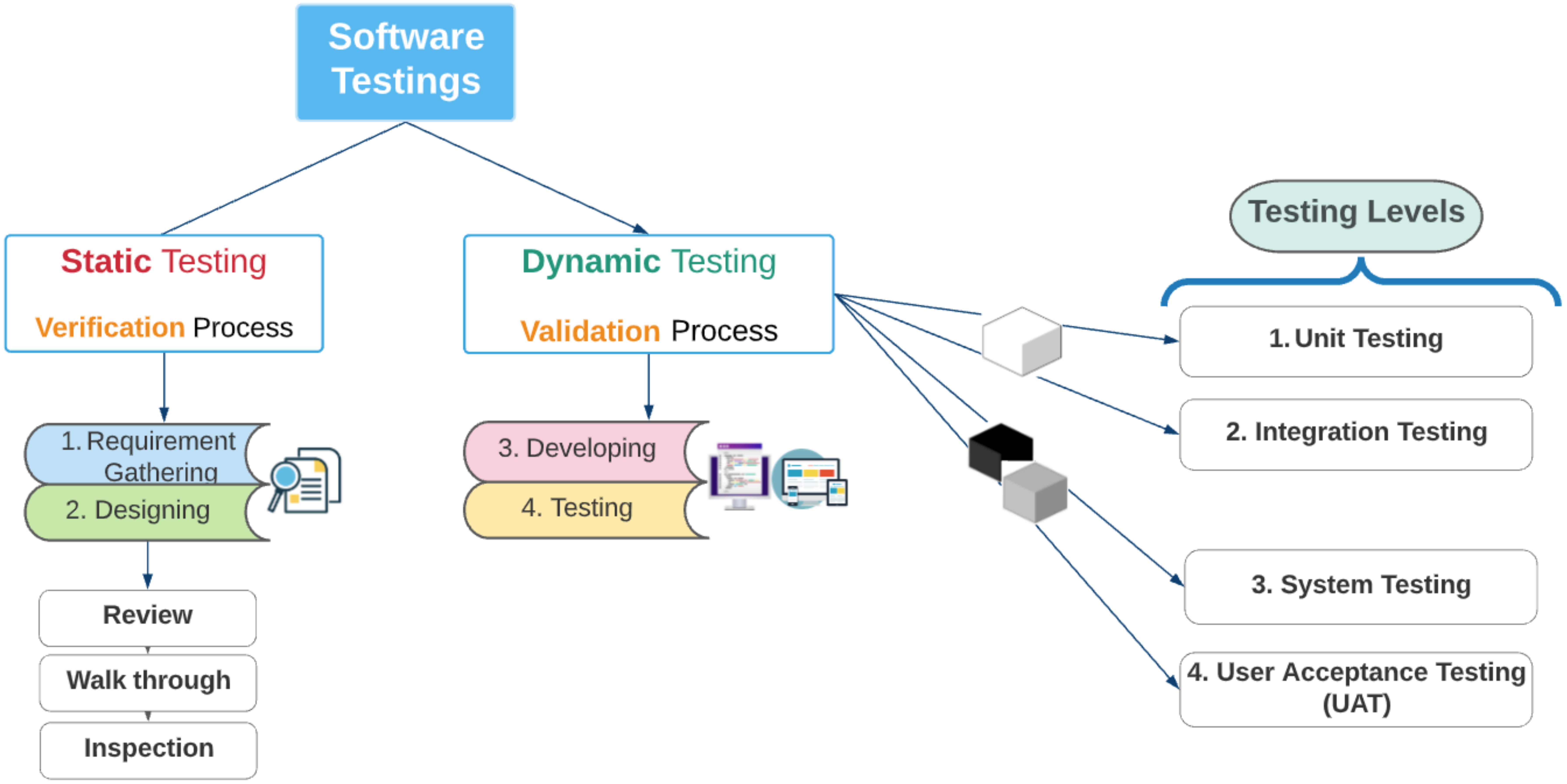
What is **Unit** Testing? **Who** performs it? In which **Environment**?

What is **Integration** Testing? **Who** performs it? In which **Environment**?

What is **System** Testing? **Who** performs it? In which **Environment**?

What is User Acceptance Testing (**UAT**)? Who performs it? In Which Environment?

CYDEO


Software Testings
Static Testing
Verification Process
Dynamic Testing
Validation Process
1. Requirement Gathering
2. Designing
3. Developing
4. Testing
Review
Walk through
Inspection
Testing Levels
1. Unit Testing
2. Integration Testing
3. System Testing
4. User Acceptance Testing (UAT)

Unit Testing

- The **first level** of testing

- Test the software's **individual unit** or **module** from the develoepr's code perspective

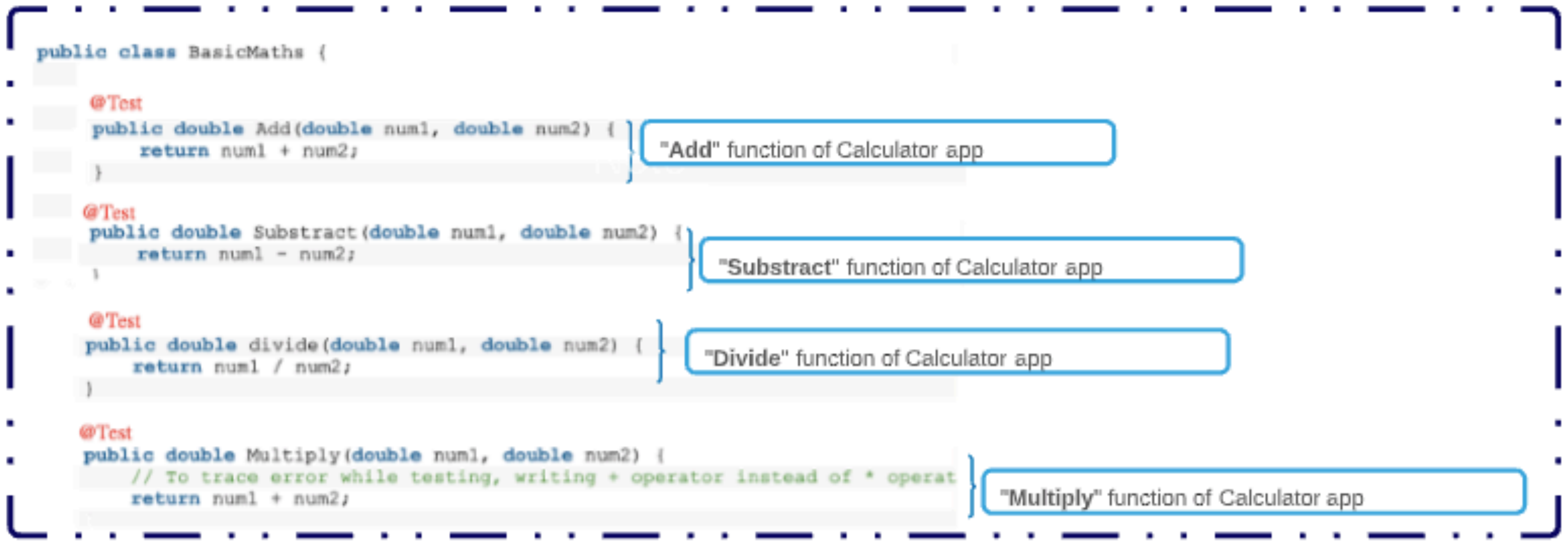
- Also called **Module** testing, or **Component** Testing

- Developers write unit tests for their code to make sure that the code works correctly. It allows developers to modify code without affecting the functionality of other units or the product as a whole.

- **Performed by the developers**

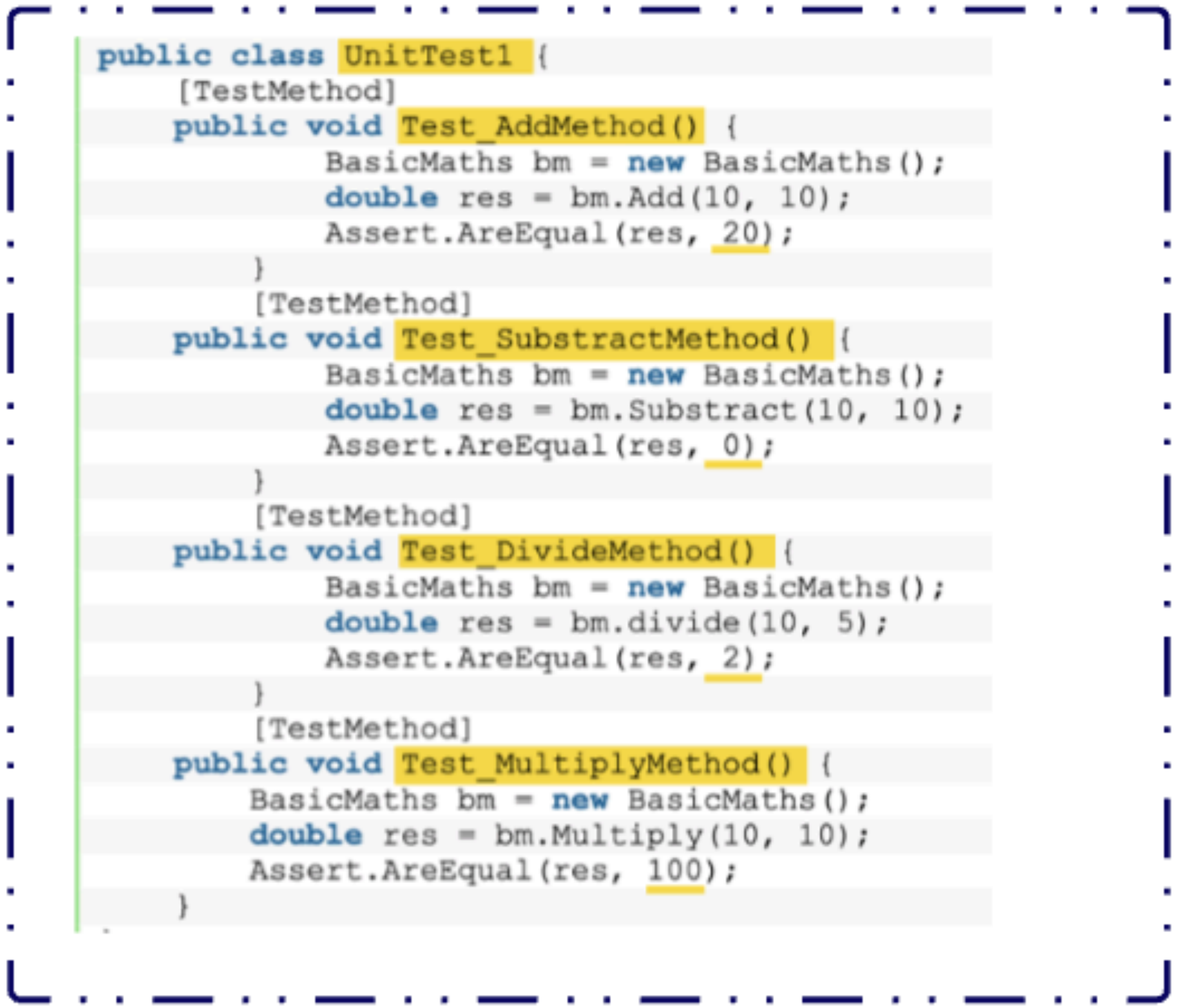
- **In Development Environment**

- **Unit testing is part of White box testing**. [Developers know the internal code knowledge when they perform unit testing.]


```
public class BasicMaths {

    @Test
    public double Add(double num1, double num2) {
        return num1 + num2;
    }
   @Test
    public double Substract(double num1, double num2) {
        return num1 - num2;
    }
    @Test
   public double divide(double num1, double num2) {
       return num1 / num2;
   }
   @Test
   public double Multiply(double num1, double num2) {
       // To trace error while testing, writing + operator instead of * operat
       return num1 + num2;
```
"Add" function of Calculator app
"Substract" function of Calculator app
"Divide" function of Calculator app
"Multiply" function of Calculator app

Developer's code to develop the app


```
public class UnitTest1 {
    [TestMethod]
    public void Test_AddMethod() {
            BasicMaths bm = new BasicMaths();
            double res = bm.Add(10, 10);
            Assert.AreEqual(res, 20);
        }
        [TestMethod]
    public void Test_SubstractMethod() {
            BasicMaths bm = new BasicMaths();
            double res = bm.Substract(10, 10);
            Assert.AreEqual(res, 0);
        }
        [TestMethod]
    public void Test_DivideMethod() {
            BasicMaths bm = new BasicMaths();
            double res = bm.divide(10, 5);
            Assert.AreEqual(res, 2);
        }
        [TestMethod]
    public void Test_MultiplyMethod() {
          BasicMaths bm = new BasicMaths();
          double res = bm.Multiply(10, 10);
          Assert.AreEqual(res, 100);
    }
```

Unit Test

CYDEO

Integration Testing

- The **Second leve**l of testing

- Test a **group of related modules to** check data transfer and connectivity between several units/modules.

- **Performed by the developers* in most companies**

- **In Development Environment**

- **Integration testing is part of White box testing***. [Developers know the internal code knowledge when they perform integration testing.]

Note: In sorne cornpanies, the senior testers/QAs perform integration testing. The cornpany teach them how to do.

System Testing

- The **Third leve**l of testing

- The software/System is **tested as a whole** from the application prespactive.

- Testers **compare** the **actual** software/**result** with the **client's expected requirement**.

- System testing **divides** into **Functioanl** and **Non-Functioanl** testings.

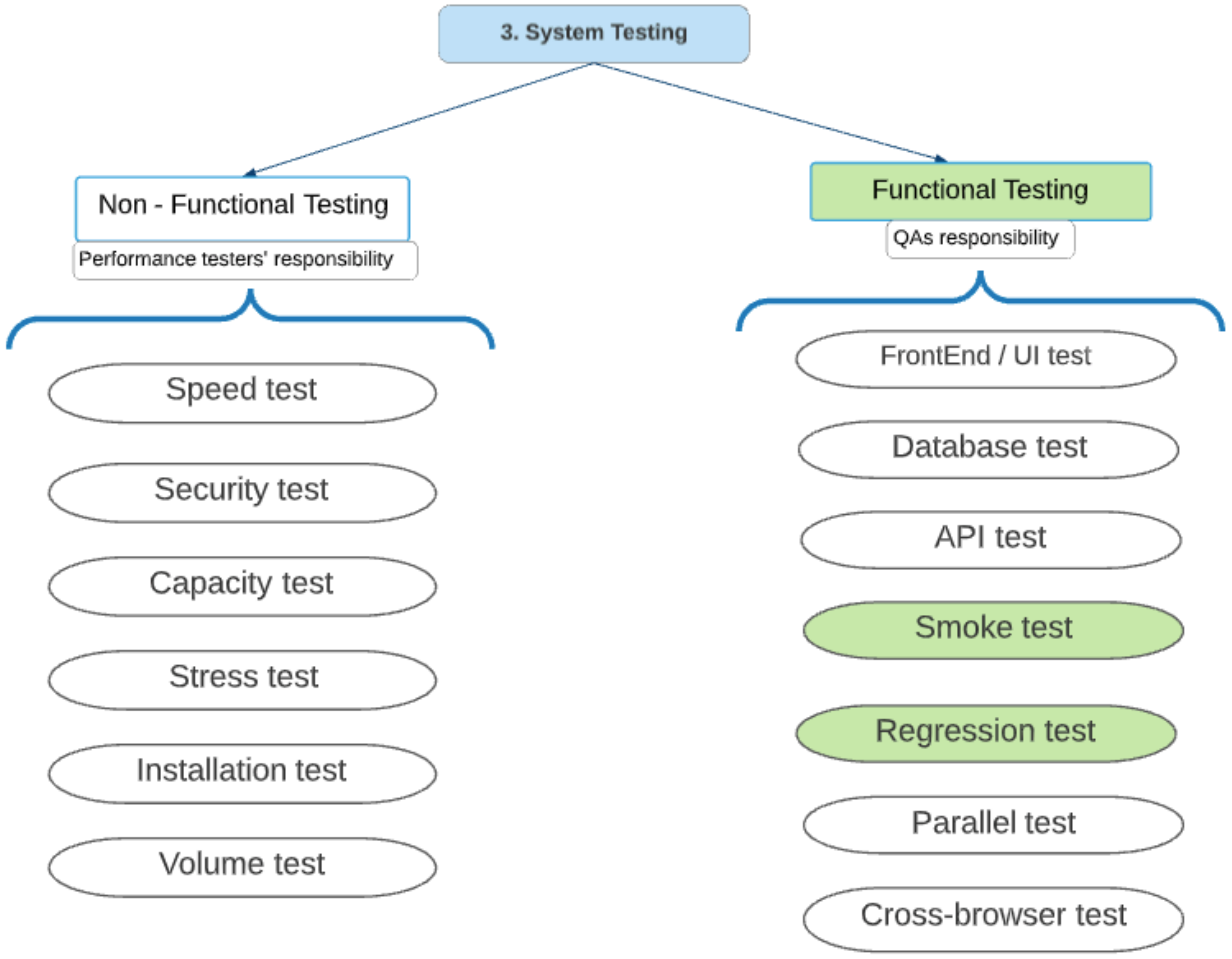
  - **Performed by the QA testers, performance testers**
  - **In QA or Test Environment**
  - **System testing is part of Black or Gray box testing**.

[Manual testers test the software without knowing the internal code - black box testing.]

[Automation testers partially know the internal code when testing the software- gray box testing .]

- System testing **divides** into **Functioanl** and **Non-Functioanl** testings.

  1. When QA/SDET **validate** the **every function** of a software as per the functional requirements, it is known as **Functional tesing**.
  2. When performance testers **validate** the performance, stress, volume etc parts of the software as per the non-functional requirements, it is known as **Non-Functional tesing**.



CYDEO

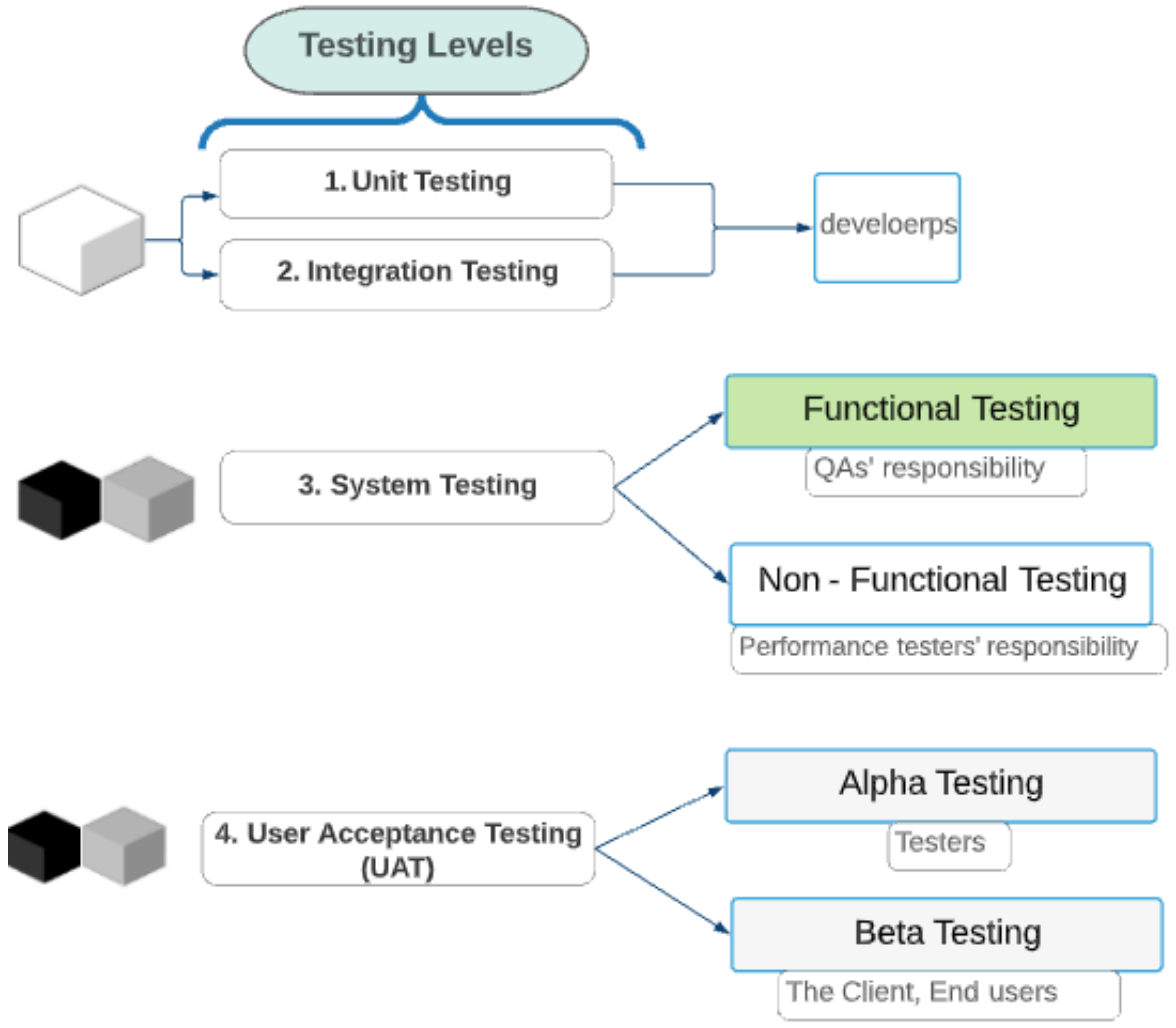
User Acceptance Testing (UAT)

- The **Fourth leve**l of testing
- UAT testing aims to evaluate whether the software i**s acceptable for release**. For UAT, PO provides special requirements based on real-world scenarios.

- UAT **divides** into **Alpha** and **Beta** testings.
  1. When UAT is carried out **by** any organization's **testers**, it is known as **alpha testing**
  2. User acceptance testing done by the client, end-users is called **beta testing**.
- **Performed by the Client, end users, testers**
- **In the Staging or Pre-Production Environment**
- **System testing is part of Black or Gray box testing**.

[The Client, end users test the software without knowing the internal code - black box testing.]

[Automation testers partially know the internal code when testing the software- gray box testing .]


Testing Levels
1. Unit Testing
2. Integration Testing
develoerps
3. System Testing
Functional Testing
QAs' responsibility
Non - Functional Testing
Performance testers' responsibility
4. User Acceptance Testing (UAT)
Alpha Testing
Testers
Beta Testing
The Client, End users

CYDEO