**🧪 Approaches to Software Testing**

Software testing ensures that code behaves as expected and meets user requirements. Common approaches include:

* **Black Box Testing**: Tests functionality without knowing internal code structure.
* **White Box Testing**: Tests internal structures or workings of an application.
* **Manual Testing**: Human testers execute test cases manually.
* **Automated Testing**: Uses tools/scripts to run tests automatically.
* **Regression Testing**: Ensures new code doesn’t break existing functionality.
* **Exploratory Testing**: Relies on tester’s intuition and experience to find issues.

**🔁 What is Test-Driven Development (TDD)?**

**Test-Driven Development (TDD)** is a software development methodology where tests are written *before* the actual code. It flips the traditional approach by making testing the foundation of development.

**✨ Key Concepts**

* **Write Tests First**: Define what the code should do before writing it.
* **Short Iterative Cycles**: Develop in small steps to maintain control and clarity.
* **Automated Unit Tests**: Focus on testing individual units of code.

**✅ Advantages**

* Ensures high test coverage and fewer bugs.
* Encourages cleaner, modular code.
* Facilitates easier refactoring.
* Builds confidence in code changes.

**🔄 The TDD Cycle: Red-Green-Refactor**

This cycle is the heartbeat of TDD. It consists of three repeating phases:

| **Phase** | **Description** |
| --- | --- |
| 🔴 **Red** | Write a test for a new feature. It should fail because the feature doesn’t exist yet. |
| 🟢 **Green** | Write the minimum code required to make the test pass. |
| 🔁 **Refactor** | Clean up the code while ensuring the test still passes. |

This cycle is repeated for every new feature or functionality.

**🧩 TDD Example: Function to Check if a Number is Prime**

Let’s walk through a simple TDD example in Python.

**Step 1: Write a Failing Test (Red)**

def test\_is\_prime():

assert is\_prime(5) == True

This test fails because is\_prime doesn’t exist yet.

**Step 2: Write Minimal Code to Pass the Test (Green)**

def is\_prime(n):

if n < 2:

return False

for i in range(2, n):

if n % i == 0:

return False

return True

Now the test passes.

**Step 3: Refactor the Code (Refactor)**

def is\_prime(n):

return n > 1 and all(n % i != 0 for i in range(2, int(n\*\*0.5) + 1))

This version is more efficient and still passes the test.

**🧠 TDD vs Traditional Testing**

| **Aspect** | **TDD** | **Traditional Testing** |
| --- | --- | --- |
| Timing | Tests written before code | Tests written after code |
| Focus | Specification and design | Defect detection |
| Code Coverage | High (often 100%) | Variable |
| Development Speed | Slower initially, faster long-term | Faster initially, riskier long-term |
| Confidence in Code | High due to constant validation | Depends on test quality |

**📚 Additional Insights**

* TDD was popularized by Kent Beck in his book *Test-Driven Development: By Example*.
* It’s often used in Agile environments and complements practices like **Continuous Integration** and **Pair Programming**.
* TDD frameworks include **JUnit** (Java), **PyTest** (Python), **RSpec** (Ruby), and **NUnit** (.NET).