**Testing in Python**

**1. What is Testing in Python?**

Testing is a critical part of software development that helps ensure that your code behaves as expected and that changes to the code don't introduce new bugs. Python provides a built-in module called unittest, but many developers prefer **pytest** because it is simpler, more powerful, and has better support for fixtures and parameterized testing.

When you write tests, you create "test functions" that check specific functionality of your code. If the tests pass, it indicates that the code works correctly for the cases you've tested.

**2. What is pytest?**

**pytest** is a testing framework for Python that allows you to write simple and scalable test cases. It can be installed via pipand is known for its ease of use and detailed output, which makes debugging easier.

**To install pytest, run:**

Pip install pytest

**3. How to Write Tests with pytest**

**Basic Structure of a Test:**

A test function typically checks one small aspect of your code. Test functions should start with the word test\_ so pytest can automatically identify them.

Here's a basic example of a test function:

**Example:**

def test\_addition():

assert 1 + 1 == 2

The test checks if 1 + 1 = 2 and if it is then the pytest will report the test to be passed.

**Running the tests:**

You can run the tests using the **pytest** command in the terminal. It will automatically find all the test functions in the current directory and run them.

**4. Testing Example Code**

Let's say you have a function that adds two numbers:

def add(a, b):

return a + b

Now, let's write a test for it using pytest:

# test\_calculator.py

from calculator import add

def test\_add():

assert add(1, 1) == 2

assert add(-1, 1) == 0

assert add(0, 0) == 0

**Explanation:**

* **Function Name**: test\_add()
  + It tests the add() function from the calculator.py file.
* **Assertions**: assert add(1, 1) == 2
  + The assert statement checks if the result of add(1, 1) is 2. If it is, the test passes. If it's not, pytest will report a failure.

To run this test, make sure both calculator.py and test\_calculator.py are in the same directory. **Then run:**

pytest test\_calculator.py

**TestCase Methods:**

|  |  |
| --- | --- |
| **Method:** | **Checks that:** |
| assertEqual(a,b) | a == b |
| assertNotEqual(a,b) | a !=b |
| assertTrue(x) | bool(x) is true |
| assertFalse(x) | bool(x) is false |
| assertIs(a,b) | a is b |
| assertIsNot(a,b) | a is not b |
| assertIsNone(x) | x is none |
| assertIn(a,b) | a in b |
| assertNotIn(a,b) | a not in b |
| assertInInstance(a,b) | Instances(a,b) |
| assertNotInstances(a,b) | Not instances(a,b) |