Code Academy

Lecturer

Rokas Slaboševičius

Unit Testing

Data

Today you will learn

01 Unit Testing

02 CICD Unit Test Sample

Unit Tests

- Unit testing is the practice of software development in which the smallest components of a program, called units or modules, are tested separately.
- These units are checked to ensure that they behave as intended by the programmer and respond correctly to input.

Unit Tests

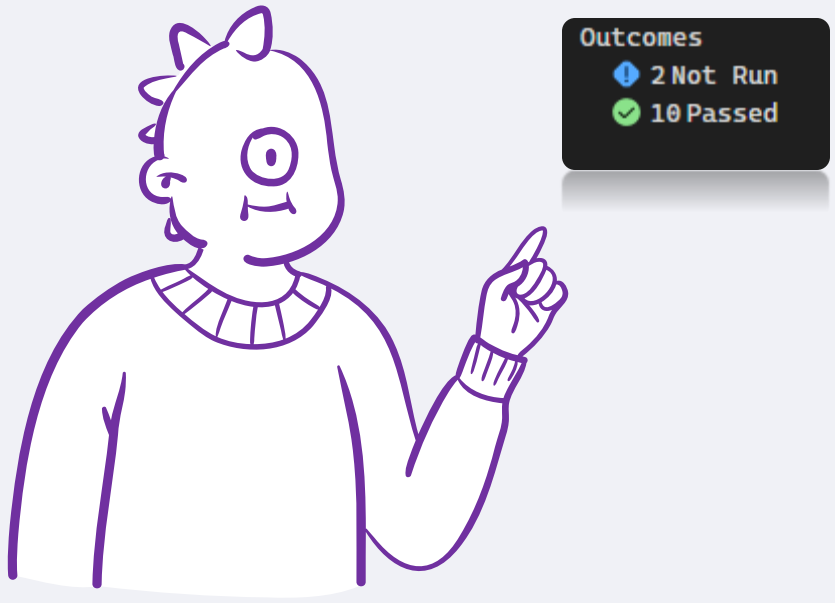
```
public static int GetTextLength(string text, bool includeLeadSpace = false)
{
    if (!includeLeadSpace)
    {
        return text.Trim().Length;
    }
    else
    {
        return text.Length;
    }
}
```

```
[TestMethod()]
| 0 references
public void GetTextLength_LeadWhiteSpacedText_ReturnsLength()
{
    // Arrange
    string fakeName = " Hello ";
    //int expected = 7;
    int expected = 5;

    // Act
    int actual = Program.GetTextLength(fakeName);

    // Assert
    Assert.AreEqual(expected, actual);
}
```

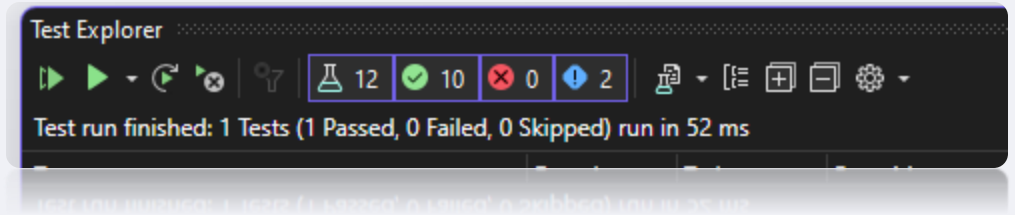
- Unit testing helps to ensure the quality of the programme. Unit testing can detect and correct bugs or defects before the application is launched or integration tested.
- This helps to ensure that each module performs its function correctly and that all modules interact properly with each other.

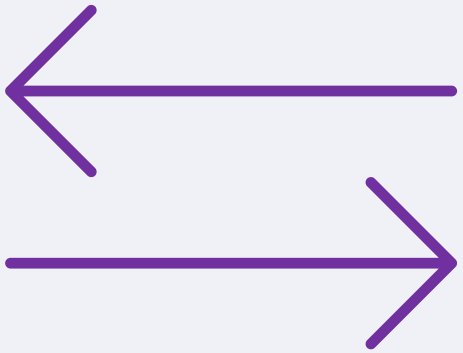
Unit Tests

- Unit testing helps simplify the search for and correction of errors.
- Each unit is tested individually, making it easier to localise and correct errors.
- Once an error is detected, efforts can be focused on that particular unit and not on the whole system.
- This saves time and resources, avoids unnecessary "debugging" and improves productivity.



Unit Tests

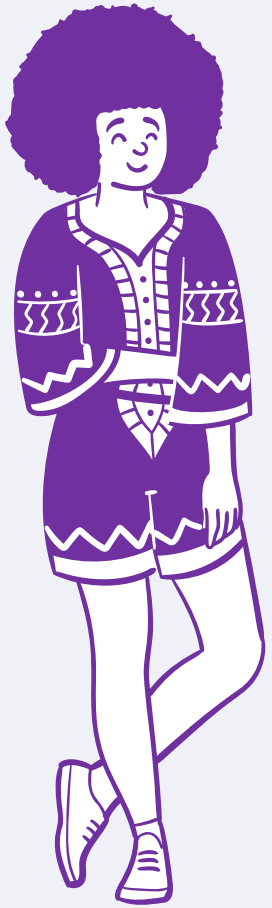


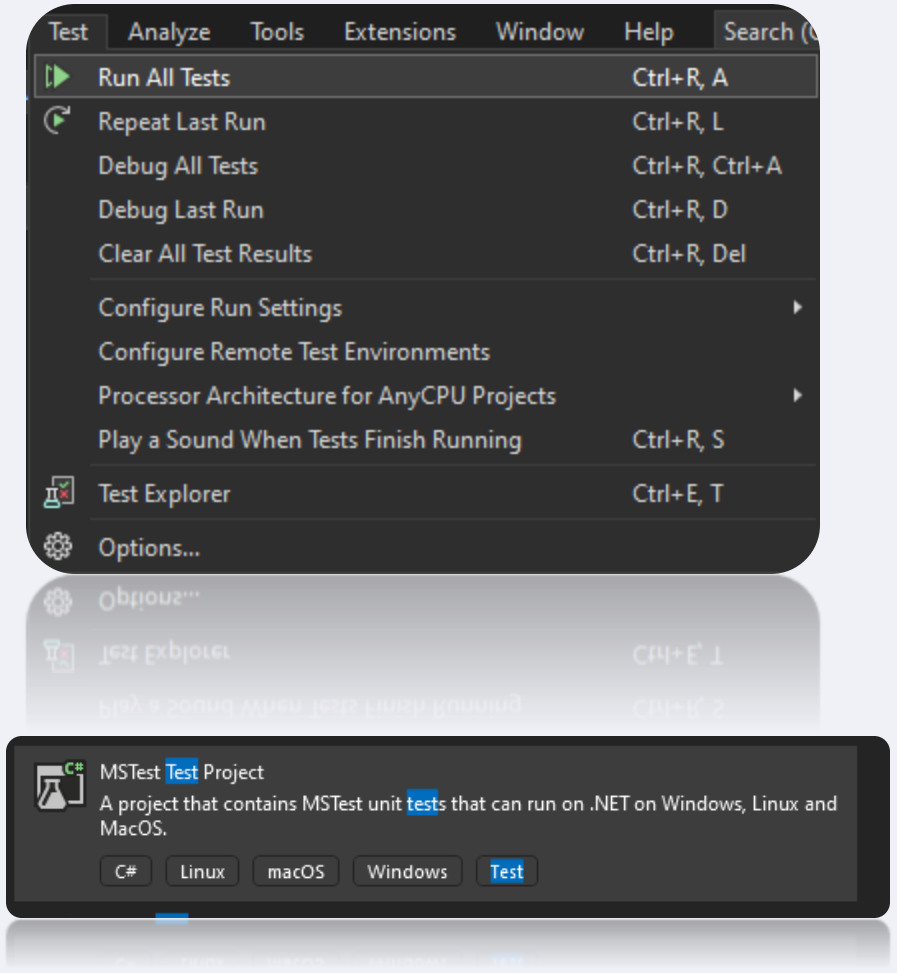


- Unit testing ensures that the code you've already written works, even after changes. With unit testing, any new code is tested before it is integrated into a larger system.
- This helps to ensure that new features or changes do not damage existing code or cause bugs.

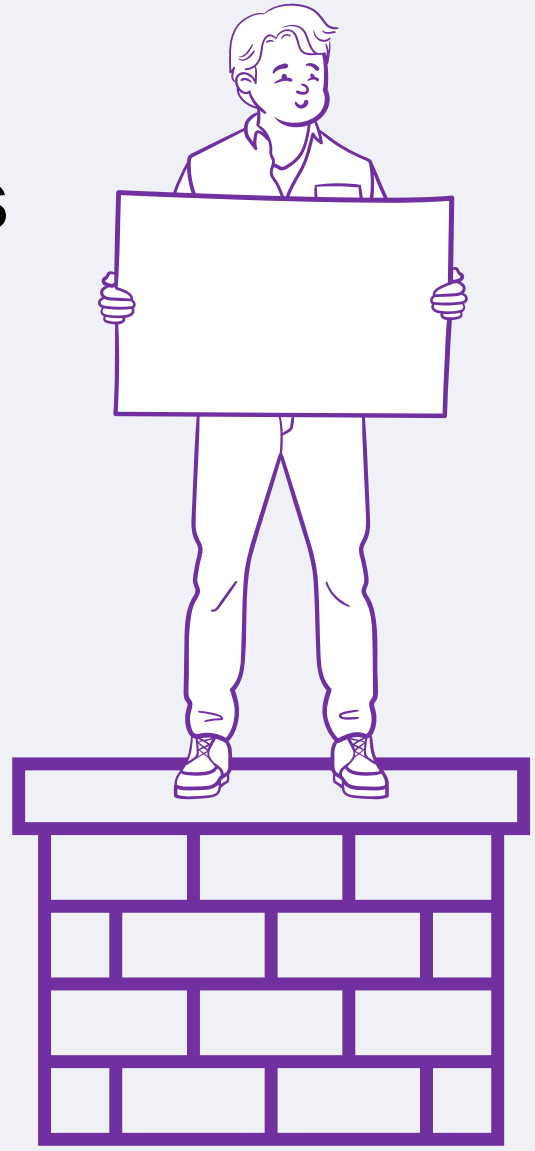
Unit Tests

- Unit testing scenarios can easily be repeated. Unit test scripts are automated so they can be easily run and repeated many times.
- This is useful for re-testing after code changes or to make sure that functionality remains stable.





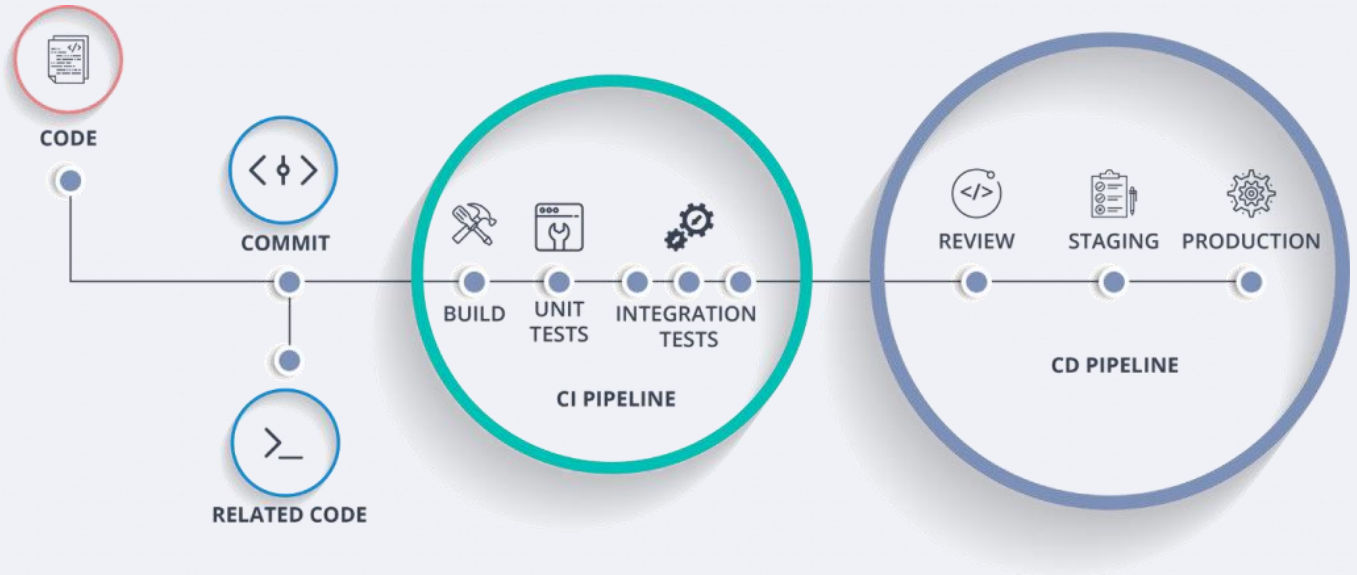
Unit Tests



- The unit testing process contributes to better coding practices.
- To test units, the code must be written in a separate state for testing.
- This encourages clear and complete code, good coding style and adherence to modularity principles.
- It also encourages developers to think about code design and compatibility with other modules.

CICD (Continuous Integration and Continuous Delivery)

- CICD is a process that aims to automate the installation, testing and delivery of software, ensuring rapid and stable product development. It involves a continuous coding, testing and delivery process where each code change is automatically tested and, if successful, also automatically delivered to the production environment.



Task 1

- Create a calculator program. Your program should be able to print menu options to the screen and should run until the letter 'q' is entered. The calculator should be able to +,-,*,/,raise a degree and extract a root. Write a unit test for each of the methods.

- Create 3 lesson (If)-task methods and create 2 tests for each of them.
- Create 4 lesson (Switch) task methods and create 2 tests for each of them.
- Create 5 lesson (String manipulation) task methods and create 2 tests for each of them.
- Create 6 lesson (While) task methods and create 2 tests for each of them. Create unit tests for the methods written in Lesson 7 (Methods).

Project No 1

- Break down the functionality of your Lesson 3 project into individual methods. When creating methods, try to put a strong emphasis on the code under test (Something should be returned from the method itself). After writing the methods, cover them all with tests. Each scenario should have at least 1 test.

References

https://learn.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2022

https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/nullable-value-types

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-forgiving