Syllabus Content:

2.4.2 Program testing:

- show understanding of ways of exposing faults in programs and ways of avoiding faults
- locate and identify the different types of errors:
  - syntax errors
  - logic errors
  - run-time errors
- correct identified errors

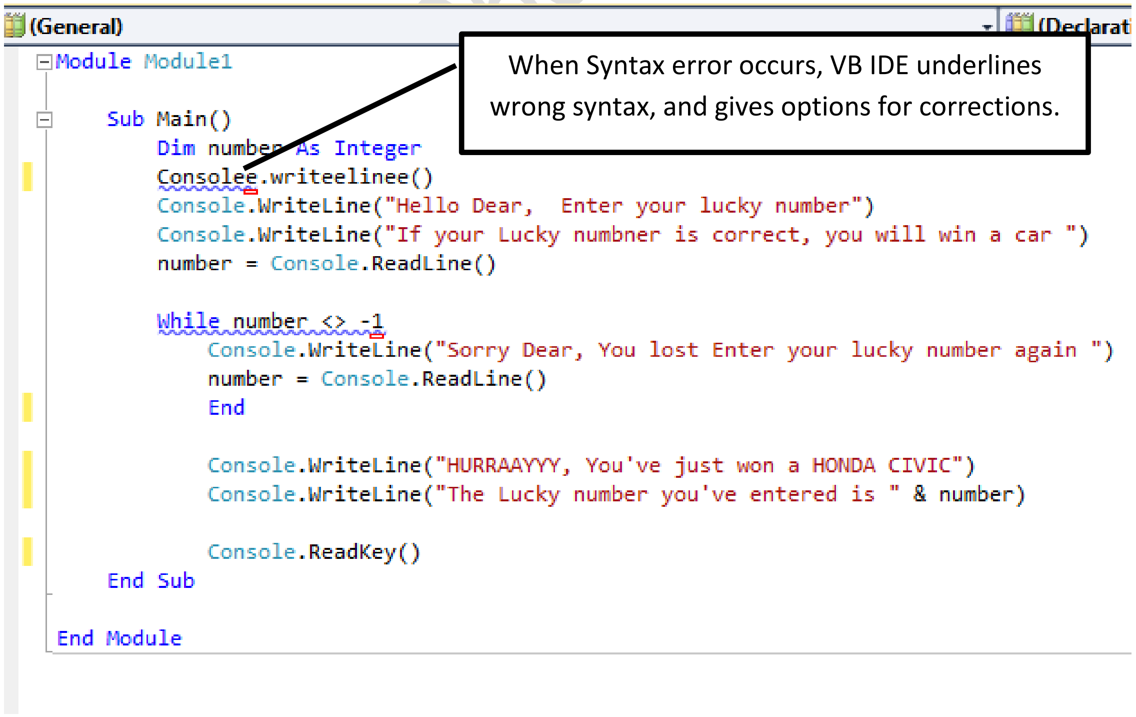
Testing strategies

Finding syntax errors is easy. The compiler/ interpreter will find them for you and usually gives you a hint as to what is wrong.

Syntax Error:

A **syntax error** is a 'grammatical' error, in which a program statement does not follow the rules of the high-level language constructs.

Syntax error: an error in which a program statement does not follow the rules of the language



```
Module Module1

    Sub Main()
        Dim number As Integer
        Consolee.writeelinee()
        Console.WriteLine("Hello Dear,  Enter your lucky number")
        Console.WriteLine("If your Lucky numbner is correct, you will win a car ")
        number = Console.ReadLine()

        While number <> -1
            Console.WriteLine("Sorry Dear, You lost Enter your lucky number again ")
            number = Console.ReadLine()
            End

            Console.WriteLine("HURRAAYYY, You've just won a HONDA CIVIC")
            Console.WriteLine("The Lucky number you've entered is " & number)

            Console.ReadKey()
    End Sub

End Module
```

When Syntax error occurs, VB IDE underlines wrong syntax, and gives options for corrections.

Much more difficult to find are **logic errors** and **run-time errors.**

A run -time error occurs when program execution comes to an unexpected ha lt or 'crash' or it goes into an infinite loop and 'freezes'.

Logical Error:

Logic error: an error in the logic of the solution that causes it not to behave as intended

Run-time Error:

Run-time error: an error that causes program execution to crash or freeze

Testing

Both of these types of errors can only be found by careful testing. The danger of such errors is that they may only manifest themselves under certain circumstances. If a program crashes every time it is executed, it is obvious there is an error.

If the program is used frequently and appears to work until a certain set of data causes a malfunction.

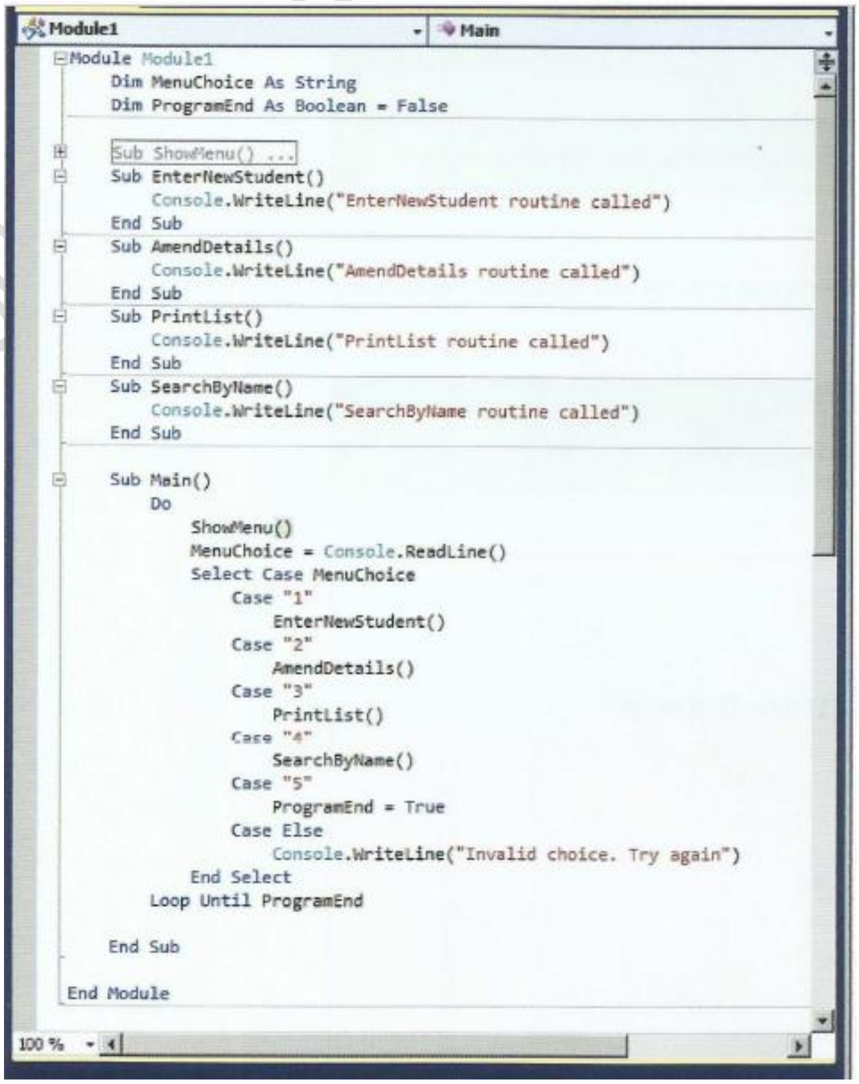
That is much more difficult to discover without perhaps serious consequences.

Stub testing

When you develop a user interface, you may wish to test it before you have implemented all the facilities.

You can write a **'stub'** for each procedure (see Figure).

The procedure body only contains an output statement to acknowledge that the call was made.
Each option the user chooses in the main program will call the relevant procedure.



```
Module Module1
    Dim MenuChoice As String
    Dim ProgramEnd As Boolean = False

    Sub ShowMenu() ...
    Sub EnterNewStudent()
        Console.WriteLine("EnterNewStudent routine called")
    End Sub
    Sub AmendDetails()
        Console.WriteLine("AmendDetails routine called")
    End Sub
    Sub PrintList()
        Console.WriteLine("PrintList routine called")
    End Sub
    Sub SearchByName()
        Console.WriteLine("SearchByName routine called")
    End Sub

    Sub Main()
        Do
            ShowMenu()
            MenuChoice = Console.ReadLine()
            Select Case MenuChoice
                Case "1"
                    EnterNewStudent()
                Case "2"
                    AmendDetails()
                Case "3"
                    PrintList()
                Case "4"
                    SearchByName()
                Case "5"
                    ProgramEnd = True
                Case Else
                    Console.WriteLine("Invalid choice. Try again")
            End Select
        Loop Until ProgramEnd

    End Sub

End Module
```

References:

- Cambridge International AS & A level Computer Science Course book by Sylvia Langfield and Dave Duddell
- Visual Basics Console Mode Editor Window from notes of Sir Majid Tahir