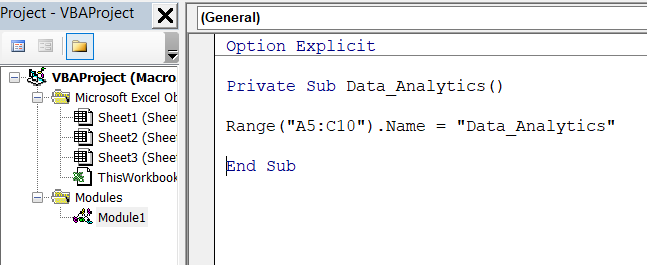
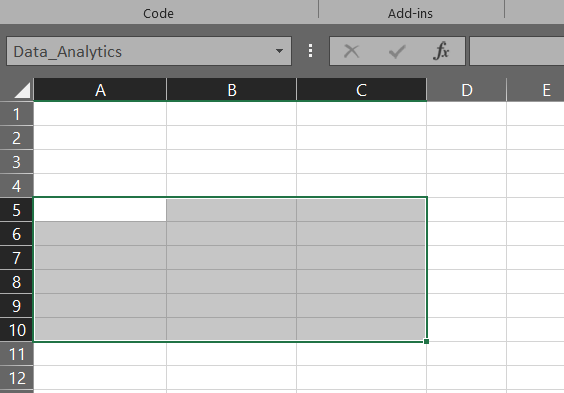
***Excel Assignment - 20***

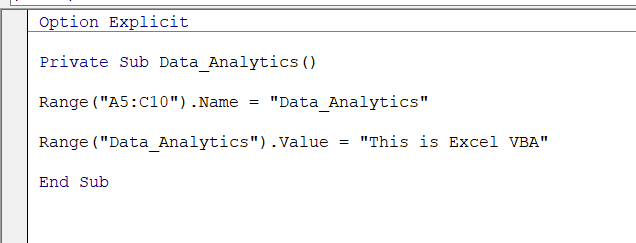
*1. Write a VBA code to select the cells from A5 to C10. Give it a name “Data Analytics” and ﬁll the cells with the following cells “This is Excel VBA”.*

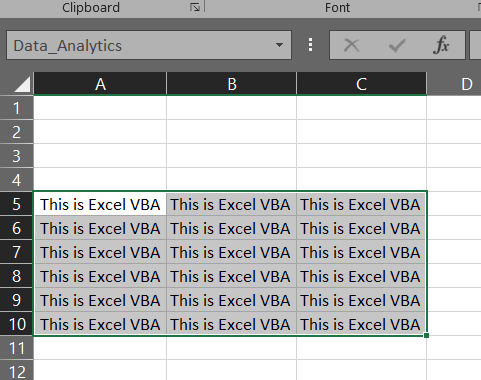


When selected “Data Analytics” as cell referance , cell A5 to C10 get selected .



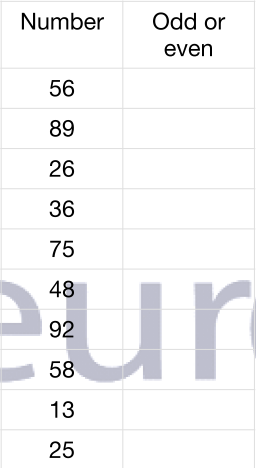
Now fill the cell “Data\_analytics ” with "This is Excel VBA".





*2. Use the above data and write a VBA code using the following statements to display in the next column if the number is odd or even*

*a. IF ELSE statement b. Select Case statement c. For Next Statement*



*a. by using If else statement :-*

*Private Sub Even\_odd()*

*Dim cell As Range*

*For Each cell In Worksheets("sheet3").Range("A2:A11")*

*If cell.Value Mod 2 = 0 Then*

*cell.Offset(0, 1).Value = "Even"*

*Else: cell.Offset(0, 1).Value = "Odd"*

*End If*

*Next cell*

*End Sub*

*b. Select Case statement*

*Private Sub Even\_odd()*

*Dim cell As Range*

*For Each cell In Worksheets("sheet3").Range("A2:A11")*

*Select Case (cell Mod 2) = 0*

*Case True*

*cell.Offset(0, 1).Value = "Even"*

*Case False*

*cell.Offset(0, 1).Value = "Odd"*

*End Select*

*Next cell*

*End Sub*

*3. What are the types of errors that you usually see in VBA?*

In Visual Basic, errors fall into one of three categories: syntax errors, run-time errors, and logic errors.

**Syntax Errors**

Syntax errors are those that appear while we write code. If you're using Visual Studio, Visual Basic checks our code as we type it in the Code Editor window and alerts us if we make a mistake, such as misspelling a word or using a language element improperly. If we compile from the command line, Visual Basic displays a compiler error with information about the syntax error. Syntax errors are the most common type of errors. We can fix them easily in the coding environment as soon as they occur.

**Run-Time Errors**

Run-time errors are those that appear only after we compile and run our code. These involve code that may appear to be correct in that it has no syntax errors, but that will not execute. For example, we might correctly write a line of code to open a file. But if the file does not exist, the application cannot open the file, and it throws an exception. We can fix most run-time errors by rewriting the faulty code or by using exception handling, and then recompiling and rerunning it.

**Logic Errors**

Logic errors are those that appear once the application is in use. They are most often faulty assumptions made by the developer, or unwanted or unexpected results in response to user actions. For example, a mistyped key might provide incorrect information to a method, or we may assume that a valid value is always supplied to a method when that is not the case. Although logic errors can be handled by using exception handling (for example, by testing whether an argument is Nothing and throwing an ArgumentNullException), most commonly they should be addressed by correcting the error in logic and recompiling the application.

*4. How do you handle Runtime errors in VBA?*

* Run-time errors are those that appear only after we compile and run our code.
* These involve code that may appear to be correct in that it has no syntax errors, but that will not execute.
* For example, we might correctly write a line of code to open a file. But if the file does not exist, the application cannot open the file, and it throws an exception.
* We can fix most run-time errors by rewriting the faulty code or by using exception handling, and then recompiling and rerunning it.

*5. Write some good practices to be followed by VBA users for handling errors*

1. Use ‘On Error Go [Label]’ at the beginning of the code. This will make sure any error that can happen from there is handled.
2. Use ‘On Error Resume Next’ ONLY when you’re sure about the errors that can occur. Use it with expected error only. In case you use it with unexpected errors, it will simply ignore it and move forward. You can use ‘On Error Resume Next’ with ‘Err.Raise’ if you want to ignore a certain type of error and catch the rest.
3. When using error handlers, make sure you’re using Exit Sub before the handlers. This will ensure that the error handler code is executed only when there is an error (else it will always be executed).
4. Use multiple error handlers to trap different kinds of errors. Having multiple error handler ensures that an error is properly addressed. For example, you would want to handle a ‘type mismatch’ error differently than a ‘Division by 0’ run-time error.

*6. What is UDF?* *Why are UDF’s used? Create a UDF to multiply 2 numbers in VBA*

* Microsoft offers us many built-in functions to speed up the work in Excel. However, we can create our functions using VBA coding, which is technically called “User-Defined Functions” (UDF).
* They are also called “custom functions” in Excel VBA.
* Even though UDF is part of our module, they are not part of our regular Subroutine in VBA.It is called a Function procedure in VBA.
* Like how we start the macro coding with the word SUB similarly, we need to start this by using the word “Function.” Sub procedure has a start and End. Similarly, the Function procedure has “Start” and “End.”

**Why are UDF’s used?**

1. Custom Functions in VBA help to simplify complex formulas. It's much easier to call a function than to write, rewrite, or copy and paste entire formulas, especially when they are frequently used.
2. UDFs can easily be copied to other workbooks.
3. We are able to do some advanced error handling within our functions since we are customizing them to be exactly how we want them.
4. The UDFs can be called from other macros, if needed.

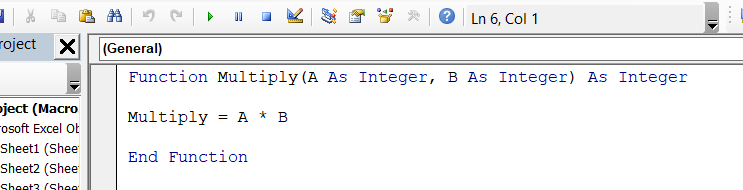
**UDF Multiplication Function to multiply 2 number :-**

1. We will start the coding with the word “Function” to create UDF.

2. we need to mention arguments here. Here, we have declared the arguments “A as Integer” and “B as Integer.” Since we multiply numerical value, we only need to assign the data type as a numerical data type.

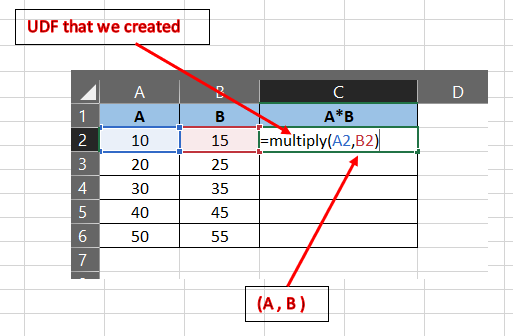
3. After declaring the arguments, we have assigned the return type of the integer too because the result given by the Function “Multiply” is also a numerical value.

4. Now, inside the function, we need to mention the formula we will use. Here, we need to use the function to start with.



5. We mentioned the formula name “Multiply” should multiply A and B.

6. We have entered a few numbers here. Open the equal sign and start typing “Multiply.” We can see the formula name appearing here. Like how we select cells, similarly, select two cells separately.



8. We will get multiplication as below :

