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”

Sub FillRange()

'Declare a range object

Dim rng As Range

'Set the range object to the desired cells

Set rng = Range("A5:C10")

'Give the range a name

rng.Name = "Data Analytics"

'Fill the cells with the text "This is Excel VBA"

rng.Value = "This is Excel VBA"

End Sub

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

Using IF ELSE statement:

Sub OddEven\_IFELSE() Dim i As Integer For i = 5 To 10 If Cells(i, 1).Value Mod 2 = 0 Then Cells(i, 4).Value = "Even" Else Cells(i, 4).Value = "Odd" End If Next i End Sub

Using Select Case statement:

Sub OddEven\_SelectCase() Dim i As Integer For i = 5 To 10 Select Case Cells(i, 1).Value Mod 2 Case 0 Cells(i, 4).Value = "Even" Case Else Cells(i, 4).Value = "Odd" End Select Next i End Sub

Using For Next Statement:

Sub OddEven\_ForNext() Dim i As Integer For i = 5 To 10 Cells(i, 4).Value = IIf(Cells(i, 1).Value Mod 2 = 0, "Even", "Odd") Next i End Sub

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

In VBA, there are three main types of errors that are commonly encountered:

Syntax errors: these occur when the code doesn’t follow the proper syntax or grammar rules of the VBA language.

Run-time errors: these occur during the execution of the code and are caused by various factors such as incorrect input values, missing data, or incorrect calculation.

Logical errors: these occur when the code works as intended but the output is not what is expected. This can be caused by incorrect logic in the code or a lack of understanding of the requirements.

It is important to identify and correct errors as they arise to ensure the code runs smoothly and produces accurate results.

4. How do you handle Runtime errors in VBA?

Runtime errors occur in VBA when an issue occurs during the execution of a VBA code or macro. The following are some of the ways to handle runtime errors in VBA:

Using the On Error statement: This statement allows you to specify what should happen in the event of an error. You can use the On Error statement to direct the code to go to a specific line in case of an error.

Using the Err object: This object is used to get more information about the error, such as the error number, error description, and source of the error.

Using the Resume statement: This statement is used to resume the execution of the code after an error has been encountered. You can use the Resume statement in combination with the On Error statement to provide specific instructions on how to handle an error.

Debugging the code: Debugging the code is a useful way to identify and fix errors in your VBA code. You can use the VBA debugger to step through your code line by line, inspect variables, and set breakpoints.

Proper error handling: Writing proper error handling code is essential for avoiding errors and improving the stability of your VBA code. Always consider the potential for errors when writing your code and plan for how to handle them.

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

errors

Number Odd or

even

56

89

26

36

75

48

92

58

13

25

Here are some good practices for handling errors in VBA:

Use Option Explicit: This forces you to declare all variables and eliminates any confusion or mistakes that can arise from implicit data typing.

Use On Error Resume Next: This statement allows you to move to the next line of code even if an error occurs. However, it's important to be cautious with this statement and to handle the error using an error handler.

Use the Debug mode: Debug mode allows you to step through your code, check variables and their values, and identify any errors.

Use error handling blocks: Error handling blocks such as Try-Catch blocks or On Error GoTo statements can be used to handle and diagnose errors.

Test your code thoroughly: Test your code thoroughly to identify any errors and debug them before releasing the code to the end-users.

Use meaningful error messages: If an error occurs, make sure that the error message is meaningful and gives the user a clear understanding of what has gone wrong.

Document your code: Documenting your code will help you understand the logic behind the code, identify errors and debug them quickly.

Stay organized: Keep your code organized and structured to make it easier to debug and maintain.

Monitor your code performance: Monitor your code performance regularly to ensure that it's running efficiently and without any errors.

Regularly back up your code: Regularly backing up your code will prevent any loss of data or work in case of a system crash or any other unexpected event.

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

numbers in VBA

UDF stands for User-Defined Function. UDFs are custom-made functions created by the user, which can be used to perform specific tasks within a VBA code. They are used to encapsulate a specific functionality that can be reused across multiple procedures and projects.

Here is an example of how to create a UDF to multiply two numbers in VBA:

Public Function MultiplyNumbers(num1 As Double, num2 As Double) As Double MultiplyNumbers = num1 \* num2 End Function

To use this UDF in a worksheet, you can enter the formula =MultiplyNumbers(A1, A2), where A1 and A2 are the cells containing the numbers to be multiplied.